HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr
Patrick Hagelkruys

Analyse von CA-Software-Systemen mit anschließender Untersuchung ausgewählter Probleme beim Design einer CA-Software

2014

Fakultät **Elektro- und Informationstechnik**

DIPLOMARBEIT

Analyse von CA-Software-Systemen mit anschließender Untersuchung ausgewählter Probleme beim Design einer CA-Software

Autor:
Patrick Hagelkruys

Studiengang:
Technische Informatik (DI (FH))

Seminargruppe:
KT10wWA-F

Erstprüfer:
Prof. Dr.-Ing. Uwe Schneider

Zweitprüfer:
DI Ramin Sabet

Mittweida, Juli 2014

Bibliografische Angaben

Hagelkruys, Patrick: Analyse von CA-Software-Systemen mit anschließender Untersuchung ausgewählter Probleme beim Design einer CA-Software, 137 Seiten, 29 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

Diplomarbeit, 2014

Satz: LaTeX

Referat

Für die Ausstellung und Verwaltung von digitalen Zertifikaten wird bei der Firma A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH eine Drittanbietersoftware eingesetzt. Sowohl aufgrund neuer Anforderungen durch Kundenwünsche als auch aus kryptoanalytischer Sicht ist eine Ablöse der eingesetzten Software durch eine Eigenentwicklung geplant.

Zu diesem Zweck werden in Kapitel 2.3 verschiedene CA-Software-Systeme analysiert und besonderes Augenmerk auf Designschwierigkeiten und deren Lösung gelegt. Es wird aus den dadurch erworbenen Erkenntnissen die Problemstellungen des Designs und der Implementierung in Kapitel 3 dargestellt. In Kapitel 4 werden verschiedene Lösungsmöglichkeiten aufgezeigt und diskutiert.

Anschließend werden im Kapitel 5 „kryptographische Bibliotheken“ auf die Verwendbarkeit in einem CA-Server-System geprüft. Die Testprogramme zur Erstellung eines Zertifikates, einer Sperrliste und einer OCSP Antwort belegen die Verwendbarkeit der jeweiligen Software.

Das Kapitel 6 beinhaltet Testszenarien, diese beschreiben Varianten die über einfache Standardtestfälle hinausgehen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AC	Attribute-Certificate, Seite 8
ADS	autonome dezentrale Systeme, Seite 40
API	Application Programming Interface, Seite 77
ASN.1	Abstract Syntax Notation One, Seite 49
authorityInfoAccess	Authority Information Access, Seite 11
CA	Certificate Authority, Seite 6
CC	Content Code, Seite 40
CRL	Certificate Revocation List, Seite 6
DAP	Dirctory Access Protocol, Seite 7
DF	Data Field, Seite 40
DOM	Document Object Model, Seite 57
DoS	Denial of Service, Seite 33
DSA	Digital Signature Algorithm, Seite 14
DTD	Document Type Definition, Seite 57
ECDSA	Elliptic Curve Digital Signature Algorithm, Seite 12
extKeyUsage	Extended Key Usage, Seite 11
HSM	Hardware Security Module, Seite 9
HTML	HyperText Markup Language, Seite 56
IANA	Internet Assigned Numbers Authority, Seite 7
Java EE	Java Enterprise Edition, Seite 13
JEE	Java Enterprise Edition, Seite 13
JSON	JavaScript Object Notation, Seite 56
LDAP	Lightweight Directory Access Protocol, Seite 7
NIST	National Institute of Standards and Technology, Seite 12
OCSP	Online Certificate Status Protocol, Seite 8
OID	Object Identifier, Seite 23
PHP	Früher: Personal Home Page Aktuell: PHP Hypertext Preprocessor, Seite 23
PKC	Public-Key-Certificate, Seite 8
PKCS	Public-Key Cryptography Standards, Seite 14
PKCS#10	Public-Key Cryptography Standards, Nr. 10, Seite 14
PKI	Public-Key-Infrastruktur, Seite 7

PKIX ..... Public-Key Infrastructure X.509, Seite 14

QC ..... Qualified Certificate, Seite 23

qcStatement ..... Qualified Certificate Statement, Seite 11

RA ..... Registration Authority, Seite 12

RFC ..... Request for Comments, Seite 9

RO ..... Registration Officer, Seite 6

RSA ..... Ron Rivest, Adi Shamir, Leonard Adleman, Seite 14

SAN ..... Subject Alternative Name, Seite 23

SHA1 ..... Secure Hash Algorithm 1, Seite 12

SHA256 ..... Secure Hash Algorithm, Hashgröße: 256bit, Seite 12

SO ..... Security Officer, Seite 7

SSL ..... Secure Sockets Layer, Seite 7

subjectAltName ... Sujbect Alternative Name, Seite 11

TBS ..... to be signed, Seite 76

TLS ..... Transport Layer Security, Seite 18

UPN ..... Universal Prinzipal Name, Seite 11

XCA ..... X Certificate and key management, Seite 14

XML ..... Extensible Markup Language, Seite 53

YAML ..... Früher: Yet Another Markup Language
Aktuell: YAML Ain' t Markup Language, Seite 53

ZDA ..... Zertifizierungsdiensteanbieter, Seite 5

V. Quellcodeverzeichnis

1 Einleitung

Für die Ausstellung und Verwaltung von digitalen Zertifikaten wird in der Firma A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH (kurz A-Trust) derzeit eine Software eingesetzt, die ursprünglich von der schwedischen Firma iD2 technologies stammt. Die Firma iD2 technologies wurde bereits mehrmals übernommen und als Teil der Firma SmartTrust bzw. heute als Nexus Technology weitergeführt.

Aufgrund der kryptoanalytischen Fortschritte der letzten Jahre ist eine Anpassung bzw. Ablösung der derzeitigen Software nötig. Vor allem durch die Kollisionsangriffe auf Hashfunktionen (vgl. [Bac09]) ist Handlungsbedarf gegeben. Durch den von der Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen veröffentlichten Algorithmenkatalog [Sch13] sind weitere Änderungen erforderlich. In diesem werden geeignete Algorithmen für die Erzeugung und Verwendung von qualifizierten Zertifikaten geregelt sowie eine Prognose für die nächsten Jahre erstellt. Auch aufgrund von Kundenanforderungen ist eine Ablösung der derzeitigen Software notwendig, da das eingesetzte Produkt die geforderten Neuerungen nicht umsetzen kann.

Von der Geschäftsführung und dem Management der A-Trust wurde entschieden, die Software durch eine Eigenentwicklung zu ersetzen. Dadurch soll eine Reduktion der Lizenzkosten stattfinden und das innerbetriebliche Know-how erweitert werden. Darüber hinaus sollen Wartung, zukünftige Kundenanforderungen oder kryptographische Anpassungen durch das Entwicklungsteam durchgeführt werden.

Zu diesem Zweck werden verschiedene CA-Software-Systeme analysiert und besonderes Augenmerk auf Designschwierigkeiten und deren Lösung gelegt. Es gilt, aus den erworbenen Erkenntnissen die Problemstellungen des Designs und der Implementierung darzustellen und verschiedene Lösungsmöglichkeiten zu besprechen. Anschließend sind kryptographische Bibliotheken auf die Verwendbarkeit in einem CA-Server-System zu prüfen und diese mit Beispielcodes zu belegen.

A-Trust GmbH

Die A-Trust ist der einzige akkreditierte Zertifizierungsdiensteanbieter in Österreich und Liechtenstein. Das Unternehmen wurde im Jahr 2000 gegründet und ist derzeit im Besitz österreichischer Banken, der Wirtschaftskammer und der Rechtsanwaltskammer.

2 Stand der Technik

Dieses Kapitel definiert die Grundbegriffe und deren Verwendung in der nachfolgenden Arbeit. Anschließend wird eine Analyse zum Stand der Technik durchgeführt.

2.1 Begriffe

2.1.1 Public-Key-Kryptographie

In [CA10, S. 12] wird Public-Key-Kryptographie beschrieben als die Verwendung von Schlüsselpaaren, welche hinreichend verschieden sind, sodass der Besitz eines Schlüssels es nicht erlaubt, den zweiten zu erlangen. Daraus folgt, dass ein Schüssel des Paars veröffentlicht werden kann während, der zweite weiterhin im Besitz des Signators bleibt. Aufgrund der Möglichkeit, einen Schlüssel zu publizieren wurde der Begriff Public-Key-Kryptographie (Öffentlicher-Schlüssel-Kryptographie) geprägt.

Eine weitere Beschreibung des Konzepts der Public-Key-Kryptographie liefert [Sch96, S. 461] wie folgt.

> „The concept of public-key cryptography was invented by Whitfield Diffie and Martin Hellman, and independently by Ralph Merkle. Their contribution to cryptography was the notion that keys could come in pairs – an encryption key and a decryption key – and that it could be infeasible to generate one key from the other [...]."

Nach eigener Interpretation ergibt sich Folgendes: „Das Konzept der Public-Key-Kryptographie wurde von Whitefield Diffie und Martin Helmann unabhängig von Ralph Merkle erfunden. Ihr Beitrag zur Kryptographie war die Idee, dass Schlüssel in Paaren verwendet werden können - ein Verschlüsselungsschlüssel und ein Entschlüsselungsschlüssel - und es nicht möglich sein darf, einen Schlüssel aus dem Anderen zu errechnen."

2.1.2 Digitale Signatur

Die digitale Signatur ist die Umsetzung einer handschriftlichen Unterschrift auf elektronischem Weg. In [CA10, Seite 14] wird dies wie folgt beschrieben.

> „A service enabled by public-key cryptography that is not easily achievable with symmetric ciphers ist the *digital signature*. This is analogous to a handwritten signature because a single entity can sign some data, but any

number of entities can read the signature and verify its accuracy.“

Das Verfahren der digitalen Signatur basiert auf der Public-Key-Kryptographie, dabei wird der Verschlüsselungsschlüssel oft als „privater Schlüssel“ oder „private key“, der Entschlüsselungsschlüssel als „öffentlicher Schlüssel“ oder „public key“ bezeichnet. Die Anwendung der Schlüssel ist in Abbildung 2.1 graphisch dargestellt.

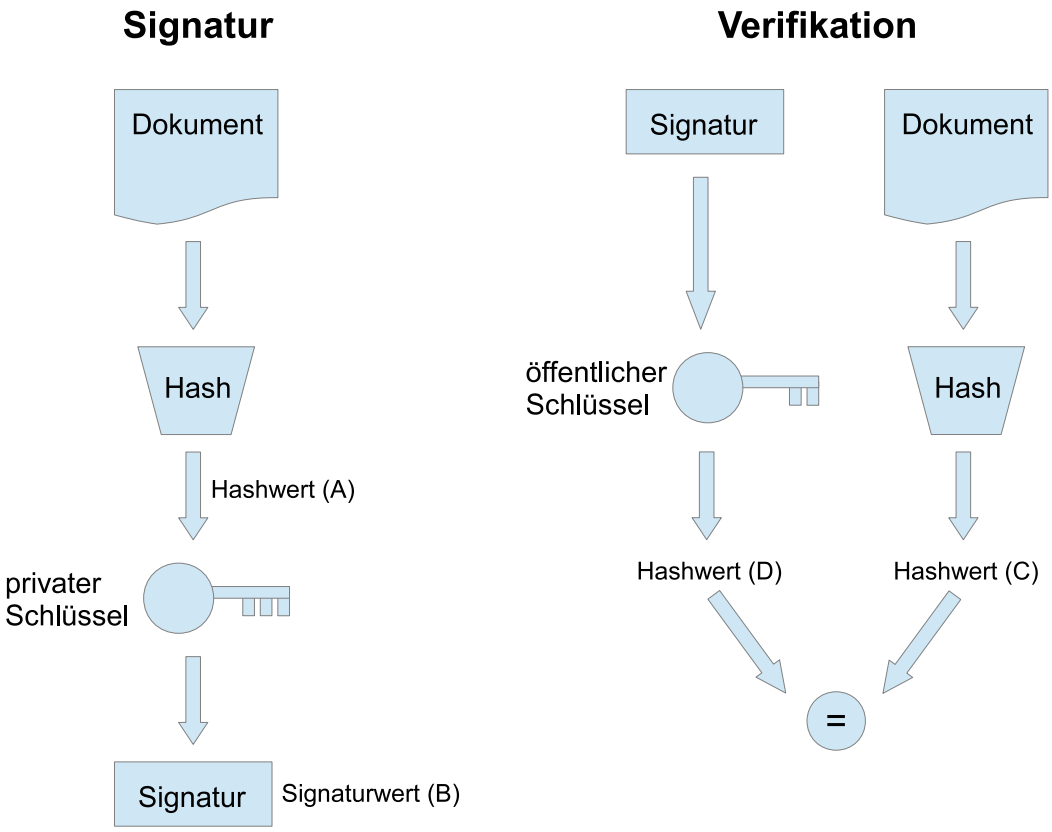


Abbildung 2.1: Ablauf Signatur und Verifikation

Zum Signieren wird auf das Dokument eine nicht rückführbare Hashfunktion angewandt, sodass der Hashwert (A) entsteht. Durch Verschlüsselung dieses Hashwertes mit dem privaten Schlüssel wird der Signaturwert (B) berechnet. Der Signaturwert mit dem öffentlichen Schlüssel und das Dokument werden veröffentlicht.

Zur Verifikation der Signatur kann sich jede Person die drei veröffentlichten Daten herunterladen. Aus dem Dokument wird analog zum Signaturablauf der Hashwert (C) berechnet. Durch Entschlüsseln des Signaturwertes mit dem öffentlichen Schlüssel wird der Hashwert (D) berechnet. Um die Gültigkeit der Signatur zu bestätigen, müssen die Hashwerte (C) und (D) verglichen werden. Sind diese identisch, ist die Signatur des Dokuments gültig.

2.1.3 Digitales Zertifikat

Ein Zertifikat ist die Verknüpfung des öffentlichen Schlüssels einer Person mit deren Personendaten. Diese Verknüpfung wird von einer vertrauenswürdigen Stelle (ZDA, Zertifizierungsdiensteanbieter) durch eine digitale Signatur (Kapitel 2.1.2) bestätigt. Der vereinfachte Aufbau eines Zertifikats ist in Abbildung 2.2 dargestellt, er enthält folgende Felder:

Version: Version des Zertifikats.
Serial Number: Eindeutige Seriennummer, die vom ZDA vergeben wird.
Algorithm Identifier: Signaturalgorithmus, mit dem der Aussteller das Zertifikat signiert hat.
Issuer: Name des ausstellenden Zertifikats.
Period of Validity: Gültigkeitszeitraum des Zertifikats.
Subject: Name der Person, welcher der öffentliche Schlüssel gehört.
Subject Public Key: Öffentlicher Schlüssel; zusätzlich wird der Algorithmus des Schlüssels angegeben.
Signature: Elektronische Signatur des Zertifikats, durchgeführt vom Aussteller (Issuer); damit kann die Gültigkeit des Zertifikats überprüft werden.

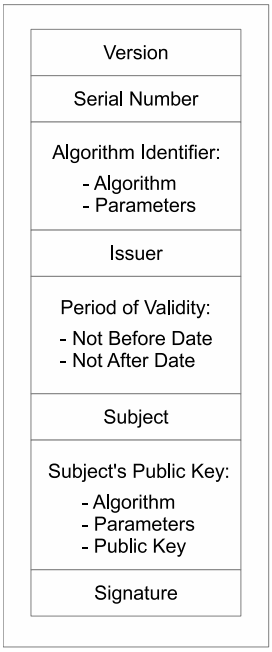


Abbildung 2.2: X509-Zertifikat, Quelle: [Sch96, S. 574]

Zusätzlich zu dem grundlegenden Aufbau aus Abbildung 2.2 kann ein Zertifikat noch Zertifikatserweiterungen enthalten. Diese sind eine Möglichkeit, dem Zertifikat weitere Eigenschaften (Attribute) hinzuzufügen, wie zum Beispiel Einschränkungen der Verwendbarkeit, Benutzerinformationen für Windows Logon, Berufsbeschreibungen, aber auch die Adresse von Sperrinformationen und Ausstellerinformationen.

2.1.4 Sperrliste, Sperrinformationen (CRL)

Sperrinformationen zu Zertifikaten werden in einem regelmäßigen Abstand (z.B. alle vier Stunden) veröffentlicht. Die Zertifikatssperrliste (CRL - Certificate Revocation List) enthält eine Liste der widerrufenen oder gesperrten Zertifikate und wird in der Regel von dem ausstellenden CA-Zertifikat signiert. Der vereinfachte Aufbau einer CRL ist in Abbildung 2.3 gezeigt. Dieser enthält folgende Felder:

Version: Format der CRL.
Algorithm Identifier: Signaturalgorithmus, mit dem der Aussteller die CRL signiert.
Issuer: Name des zugehörigen Zertifikats.
Period of Validity: Gültigkeitszeitraum der CRL.
List of Revoked Certificates: Liste der widerrufenen bzw. gesperrten Zertifikate mit Grund und optionalem Datum.
Signature: Elektronische Signatur der CRL, durchgeführt vom Aussteller (Issuer). Damit kann die Gültigkeit der Sperrliste überprüft werden.

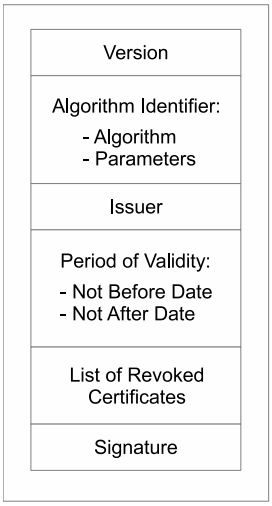


Abbildung 2.3: X509-Zertifikatssperrliste, angelehnt an den Aufbau des Zertifikats aus [Sch96, S. 574]

2.1.5 Registration Officer (RO) / Security Officer (SO)

Registration Officer (RO)

Ein Registration Officer hat die Rechte, Zertifikate für andere Personen auszustellen oder zu widerrufen. Die technischen Beschreibung der SmartTrust-CA [RWS02, S. 13] beschreibt ihn wie folgt:

> „A Registration Officer has the authority to request certification of end-users and revoke the certificates of end-users. Also, ROs can create and execute

> batch smart card production orders. ROs can be constrained to performing only certain functions such as revocation. The RO may also be restricted to issuing and revoking certificates within a specified subject name space.“

Security Officer (SO)

Ein Security Officer hat die Rechte eines Registration Officers und kann zusätzlich Änderungen an der Konfiguration der CA-Software durchführen. Die technische Beschreibung der SmartTrust-CA [RWS02, Seite 13] lautet so:

> „A Registration Officer has the authority to request certification of end-users and revoke the certificates of end-users. Also, ROs can create and execute batch smart card production orders. ROs can be constrained to performing only certain functions such as revocation. The RO may also be restricted to issuing and revoking certificates within a specified subject name space.“

2.1.6 LDAP - Lightweight Directory Access Protocol

LDAP ist ein Protokoll des Zugriffs auf Verzeichnisse über eine TCP/IP-basierte Kommunikation und ist als Subset des X.500 Dirctory Access Protocol (DAP) entstanden (vgl. [CA10, S. 226] und [Ser06]). Ein LDAP-Verzeichnisdienst wird im Public-Key-Infrastruktur (PKI) Umfeld für die Veröffentlichung von ausgestellten Zertifikaten und Sperrinformationen (CRL) verwendet.

Der Nachteil dieses Protokolls ist der zusätzlich benötigte TCP/IP-Port für den Verbindungsaufbau des Clients. Dem LDAP-Protokoll wurden von der Internet Assigned Numbers Authority[1] (IANA) die Portnummern 389 und 636 (SSL) zugewiesen. Diese Ports sind in den meisten Firmennetzwerken nicht offen. Daher kann eine Abfrage der Sperrinformationen in vielen Fällen nicht durchgeführt werden bzw. muss dieser Port freigegeben werden.

2.1.7 Attributszertifikat

Ein Attributszertifikat ist ähnlich einem Zertifikat (vgl. Kapitel 2.1.3) aufgebaut, enthält jedoch keinen öffentlichen Schlüssel. Attributszertifikate können erweiterte Eigenschaften (Attribute) zu einem Zertifikat hinzufügen. Einige Beispiele sind Gruppenzugehörigkeiten, Rollen, Sicherheitsüberprüfung oder andere Anmeldeinformationen. Eine Beschreibung des Unterschieds zwischen öffentlichen Schlüsselzertifikaten nach Kapitel 2.1.3 und Attributszertifikaten liefert der zugehörige Standard RFC 5755 (s. [SF10]):

[1] http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=ldap

> „A PKC [public key certificate; Anmerk. d. Verf.] can be considered to be like a passport: it identifies the holder, tends to last for a long time, and should not be trivial to obtain. An AC [attribute certificate; Anmerk. d. Verf.] is more like an entry visa: it is typically issued by a different authority and does not last for as long a time. As acquiring an entry visa typically requires presenting a passport, getting a visa can be a simpler process.“

Daraus ergibt sich nach eigener Interpretation:
Ein Zertifikat eines öffentlichen Schlüssels kann mit einem Ausweis verglichen werden. Es identifiziert den Besitzer, hält mehrere Jahre und ist kompliziert in der Ausstellung. Ein Attributszertifikat ist ähnlich einem Visum. Es wird normalerweise von einer anderen Stelle ausgestellt und besitzt eine kürzere Gültigkeit. Üblicherweise wird ein Ausweis benötigt, um ein Visum zu bekommen. Die Erstellung eines Visums ist ein einfacherer Prozess.

2.2 Beschreibung einer CA-Software

Eine Certificate Authority Software oder kurz CA-Software ist eine Software zum Erstellen, Widerrufen und Verwalten von digitalen Zertifikaten. Wie in [Sta01, Kapitel 4.7.4 – PKI-Dienste] und [Hag13, Kapitel 2.4 – Kernaufgaben der CA-Software] beschreiben, stellt eine CA-Software diverse PKI-Dienste (bzw. Kernaufgaben) zur Verfügung. Zusammengefasst ergeben sich die nachfolgend beschriebenen Aufgaben:

Sichere Identifikation und Registrierung: Bei der Ausstellung eines Zertifikates wird der Kunde von einem Registration Officer (kurz RO) durch einen Lichtbildausweis identifiziert. Anschließend wird ein Zertifikat mit den Personendaten des Kunden erstellt und diesem übergeben.

Ausstellung von Zertifikaten: Ein Zertifikat ist die Verknüpfung einer digitalen Signatur mit einer Person. Damit dieser Zusammenhang für andere Teilnehmer erkenntlich ist, bestätigt die CA das erstellte Zertifikat mit ihrer eigenen digitalen Signatur. Ein CA-Zertifikat wird über eine eigene Managementschnittstelle im Vieraugenprinzip erstellt.

Sperrung und Widerruf von Zertifikaten: Zertifikatsinhaber können ihr Zertifikat zu einem beliebigen Zeitpunkt sperren oder widerrufen. Die Durchführung wird durch einen RO bestätigt.

Verwaltung von Zertifikaten: In regelmäßigen Abständen wird für jedes CA-Zertifikat eine Sperrliste (CRL) erstellt und veröffentlicht. Diese enthält alle gesperrten und widerrufenen Zertifikate. Zusätzlich können Zertifikatsstatusabfragen über Online Certificate Status Protocol (kurz OCSP) durchgeführt werden.

Zu den erwähnten Aufgaben/PKI-Diensten beinhaltet eine CA-Software zusätzliche Funktionen, die nicht als extern verwendbare Dienste verfügbar sind oder als eigenständige

Aufgabenbereiche identifiziert werden können. Die nachfolgenden Absätze beschreiben einzelne Bereiche, welche für die Funktionalität und Umsetzung oben genannter Aufgaben relevant sind.

Oftmals umfasst ein solches System die Integration spezieller Kryptographie-Hardware, wie Hardware-Sicherheitsmodul (HSM – Hardware Security Module) und Smartcards. Einerseits dient es der Aufbewahrung von privaten Schlüssel, andererseits zur Authentifizierung der Anwender. Diese Hardwarekomponenten werden entweder über eine Standard-Schnittstelle wie z.B. PKCS#11 (s. [RSA09]) angesprochen oder müssen über spezielle Erweiterungsmodule für die jeweilige Hardware eingebunden werden.

Die Erstellung von Zertifikaten basiert zumeist auf Zertifikatsvorlagen, oftmals auch Zertifikatsregeln oder, wie bei Microsoft-Produkten üblich, Policy genannt. Obwohl die Grundstruktur eines digitalen Zertifikats wie in Abbildung 2.2 im RFC 5280 (s. [DC08]) standardisiert ist, wird der Inhalt der meisten Felder über die Zertifikatsvorlagen bestimmt oder beeinflusst. Flexibilität und Einsatzmöglichkeit einer CA-Software zeigen sich jedoch bei der Konfiguration von Zertifikatserweiterungen. Theoretisch ist es möglich, jede beliebige Unterstruktur in einer Zertifikatserweiterung umzusetzen. Diese Flexibilität in Form von Zertifikatsvorlagen zu bringen, ist eine der größeren Herausforderungen beim Design der CA-Software und wird im Zuge der Analyse in Kapitel 2.3 näher betrachtet.

Die Ausprägungen von CA-Software-Systemen variieren von vollautomatischen Server-Systemen über Einzelplatzversionen bis hin zu Kommandozeilen-Scripts. Je nach Einsatzgebiet hat jedes System seine Vorteile und Nachteile und ist auf einen bestimmten Einsatzzweck abgestimmt. Obwohl explizit ein CA-Server-System entwickelt werden soll, ist auch die Analyse der anderen Systeme im Hinblick auf Designschwierigkeiten relevant. Vor allem wird durch die Unterschiede dieser Systeme eine bessere Definition der angestrebten Lösung möglich.

Ein weiterer wichtiger Aspekt ist der Grad der benötigten Benutzerinteraktion zum Erstellen oder Verwalten von Zertifikaten und Einstellungen. Dieser ist ein wichtiges Indiz für die Sicherheit, Vertrauenswürdigkeit und Manipulationsmöglichkeiten des Gesamtsystemes. Das Ausstellen oder Widerrufen eines Zertifikats unabhängig davon, ob es sich dabei um ein Endbenutzer- oder ein Zwischeninstanzzertifikat handelt, wird in vielen Systemen mittels einer PIN-Eingabe über die Tastatur gesichert. Weitere Authentifizierungsmöglichkeiten reichen von Windows-Domain-Authentifizierung bis hin zu der Bestätigung im Vieraugenprinzip mittels Hardwaretokens (Smartcard, RSA-Token).

Die benötigte Benutzerinteraktion spiegelt sich auch im nächsten Punkt, dem Rollenkonzept, wider. Das Rollenkonzept einer CA-Software beschreibt, welche unterschiedlichen Rollen in der CA-Software definiert sind und wie deren Zusammenspiel beschaffen sein muss, um eine bestimmte Aktion auszuführen. Bei den einfachsten Lösungen wird ei-

ne Rolle des „Administrators“ vergeben, welcher Zugriff auf den privaten Schlüssel des CA-Zertifikats hat und somit alle Funktionen ausführen kann.

2.3 Analyse von CA-Software Systeme

Im folgenden Kapitel werden bestehende CA-Software-Systeme untersucht. Hier liegt das Hauptaugenmerk auf den folgenden Punkten:

Kategorisierung nach Automatisierbarkeit Wie in Kapitel 2.2 beschrieben, gibt es mehrere Ausprägungen von CA-Software-Systemen. In diesem Punkt wird in die folgenden drei Kategorien eingeteilt:

- Server-System (Web-Interface)
- Einzelplatzsystem
- Kommandozeilen-Script

Die zu entwickelnde Software soll ein autonomes Server-System sein. Für dieses unterscheiden sich einige Anforderungen im Vergleich zu Einzelplatzsystemen oder Kommandozeilen-Scripten. Daher ist auf Basis der Kategorisierung zu entscheiden, inwieweit die realisierten Funktionen auf das Zielsystem umgesetzt werden können. Zum Beispiel wird bei einem Einzelplatzsystem für jede Aktion ein Passwort für den CA-Schlüssel benötigt. Dieses Verhalten ist jedoch für ein autonomes System nicht praktikabel. In anderen Bereichen, wie den Zertifikatsformaten, sind keine Unterschiede zwischen Einzelplatzsystem, Server-System und Kommandozeilen-Scripten vorhanden.

Serverkomponenten, Lastverteilung Bei Server-Systemen werden zusätzlich die Aufteilung der einzelnen Komponenten und deren Kommunikationsschnittstellen untereinander betrachtet. Zusätzlich werden die vom System zur Verfügung gestellten Möglichkeiten zur Lastverteilung untersucht.
Für die weitere Betrachtung der Aufteilung der Komponenten in Kapitel 4.1 und die Betrachtung zu Ausfallszenarien und Lastverteilung in Kapitel 4.2 werden bereits Lösungsansätze untersucht.

Zertifikatsregeln, Zertifikatsvorlagen Hier wird gefragt, welche Art der Modifikation der Zertifikatsgrundstruktur die Software bietet und welche Erweiterungen mit einfachem Aufwand eingefügt werden können, aber auch, welche Möglichkeiten es gibt, Erweiterungen einzufügen, die vom Standardprogramm nicht unterstützt werden.
Im Hinblick auf die zu erstellenden Testzertifikate und auf Kapitel 4.3 ist eine Analyse der Zertifikatsregeln, -formate und -vorlagen, sowie deren Vorteile und Nachteile, erforderlich.

Rollenkonzept Zu fragen ist, welche Rollen das System vorsieht und wie diese unterschieden werden und welche Authentifizierungsmöglichkeiten für die einzelnen

Rollen vorgesehen sind.
Wie bereits in Kapitel 2.2 beschrieben, ist das Rollenkonzept ein wichtiges Indiz für die Sicherheit und Vertrauenswürdigkeit des Gesamtsystems. Durch das Rollenkonzept wird unter anderem festgelegt, welche Interaktion notwendig ist, um den privaten Schlüssel des Stammzertifikates anzuwenden.

Testzertifikate Der wichtigste Teil der Analyse ist die Erstellung von Zertifikaten nach den Vorlagen aus Anhang A. Dazu sind zuerst ein Wurzelzertifikat und eine Zertifikatswiderrufsliste (CRL) zu generieren.

Die nachfolgenden Zertifikatsvorlagen wurden so gewählt, dass eine möglichst große Abdeckung der derzeit eingesetzten Zertifikatserweiterungen besteht. Die Erstellung der Testzertifikate zeigt die Möglichkeiten, die durch die jeweiligen Zertifikatsformate, -regeln gegeben sind. Diese Erkenntnisse sind für den Vergleich in Kapitel 4.3 notwendig ebenso wie für die Entwicklung eigener Zertifikatsformate.

Bei den zu erstellenden Testzertifikaten wird das Zertifikat-Subject (Antragsteller) mit den Vorlagenzertifikaten verglichen. Als schwierigstes Feld wird die `serialNumber` angesehen, da es sich hierbei um einen seltenen Eintrag handelt. Die Zertifikatserweiterungen `keyUsage`, `basicConstraints`, `certificatePolicies` und `authorityInfoAccess` sollten in jeder Standardsoftware abgedeckt sein. Die Zertifikatserweiterung `crlDistributionPoints` sollte ebenso möglich sein, wobei die Angabe einer LDAP-URL vielleicht zu Problemen führen kann.

Zertifikat mit mehreren Domainnamen im Subject Alternative Name Zusätzlich zu den Zertifikatsfeldern, die im vorangehenden Absatz beschrieben wurden, ist hier die Zertifikatserweiterung `subjectAltName` essentiell.

Zertifikat mit UPN im Subject Alternative Name Bei diesem Zertifikat ist der Eintrag `extKeyUsage`, vor allem aber der Wert `smartcardLogon` wichtig. Dieser Wert ist eine spezielle Erweiterung von Microsoft und daher oftmals nicht in den Standardeinstellungen enthalten. Als zweiter wichtiger Punkt ist die Zertifikatserweiterung `subjectAltName` mit dem Wert `otherName` bzw. `universalPrinzipalName` (UPN) wichtig. Auch hierbei handelt es sich um eine Microsoft-Erweiterung, die oftmals nicht im Standardumfang von CA-Software-Systemen enthalten ist.

Zertifikat mit Qualified Certificate Statement Bei diesem Testzertifikat wird die Zertifikatserweiterung `qcStatement` eingefügt. Die Erweiterung selbst ist im RFC 3039 in Kapitel 3.2.5 (s. [SS01]) beschrieben.

Zertifikat mit Dienstleistereigenschaft Die Dienstleistereigenschaft ist eine Zertifikatserweiterung speziell für den österreichischen Raum. Diese Erweiterung wird vermutlich von keiner Software im Standardumfang unterstützt. Gleichzei-

tig steht dieses Testzertifikat stellvertretend für alle Zertifikatserweiterungen, die der CA-Software nicht bekannt sind.

Algorithmenunterstützung Hier ist zu fragen, welche Kryptographiealgorithmen von der Software unterstützt werden. Auf der Basis der aktuell ausgegebenen Zertifikate werden RSA-Schlüssel bis zu 4096 Bit und ECDSA-Schlüssel entsprechend der NIST-Kurven P192 und P256 verwendet. Die benötigten Hash-Algorithmen sind SHA1 und SHA256.

2.3.1 Smarttrust CA

Aktuell wird zur Ausstellung von Zertifikaten eine Software eingesetzt, die ursprünglich von der schwedischen Firma iD2 technologies stammt. Die Firma iD2 technologies wurde bereits mehrmals übernommen und als Teil der Firma SmartTrust bzw. heute als Nexus Technology weitergeführt.

Das Server-System ist in drei Komponenten aufgeteilt, eine Certificate Authority (CA), einen Distribution Manager (DM) und einen OCSP-Server. Die Certificate Authority hat die Aufgabe, Zertifikate auszustellen und zu verwalten, der Distribution Manager ist für die Veröffentlichung der Zertifikate zuständig. Die Kommunikation der beiden Systeme basiert auf einer „Java remote method invocation“Technologie. Der OCSP-Server ist eine eigenständige Komponente und benötigt Zugriff auf die aktuellen Sperrlisten in einem LDAP-Server.

Zertifikatsvorlagen werden in zwei Kategorien eingeteilt, in den Bereich Zertifikatsprozedur und in den Bereich Zertifikatsformat. Die Zertifikatsprozedur enthält Informationen, die durch die verwendete CA definiert werden, und ist die Verknüpfung zwischen CA-Zertifikatsformat und Veröffentlichungsregeln. Diese Prozeduren werden in der Datenbank abgespeichert. Das Zertifikatsformat enthält Zertifikatsregeln, welche die Zertifikatsfelder befüllen, verschieben oder löschen können. Die Regeln und Formate werden im Dateisystem im INI-Datei-Format abgelegt.

Die CA-Software ist ein Server-System, das über Registration Authorities (RA) verwaltet wird. Das Rollenkonzept beinhaltet Registration Officer (RO) und Security Officer (SO). Ein Registration Officer aktiviert Benutzerzertifikate und kann diese sperren bzw. widerrufen. Einem Security Officer ist es möglich, Einstellungen an der CA-Software vorzunehmen und weitere Registration Officers anzulegen.

Die Smarttrust CA wurde zum Erstellen der Testzertifikate verwendet, die als Referenz für die Analyse der weiteren CA-Software-Systeme gelten.

2.3.2 EJBCA PKI CA

Die EJBCA ist eine Open-Source-CA-Software und unter `www.ejbca.org` erreichbar. Nachfolgend wird die Beschreibung der Software von der Homepage [EJB14b] wiedergegeben:

> „EJBCA® is an enterprise class PKI Certificate Authority software, built using Java (JEE) technology. It is a robust, high performance, platform independent, flexible, and component based CA to be used stand-alone or integrated in other JEE applications.
>
> An enterprise class PKI CA, EJBCA can be used to build a complete PKI infrastructure for your organization. If you only want to issue a few single certificates for testing, there are probably options that will get you started quicker, but if you want a serious PKI CA we recommend EJBCA.“

Die Komponenten der EJBCA teilen sich in CA-Server, OCSP-Server und RA-Server. Ein Architekturvorschlag des EJBCA-Teams ist in Abbildung 2.4 dargestellt, wobei die Kommunikation zwischen den einzelnen Komponenten asynchron über gemeinsame Datenbanken gelöst wird.

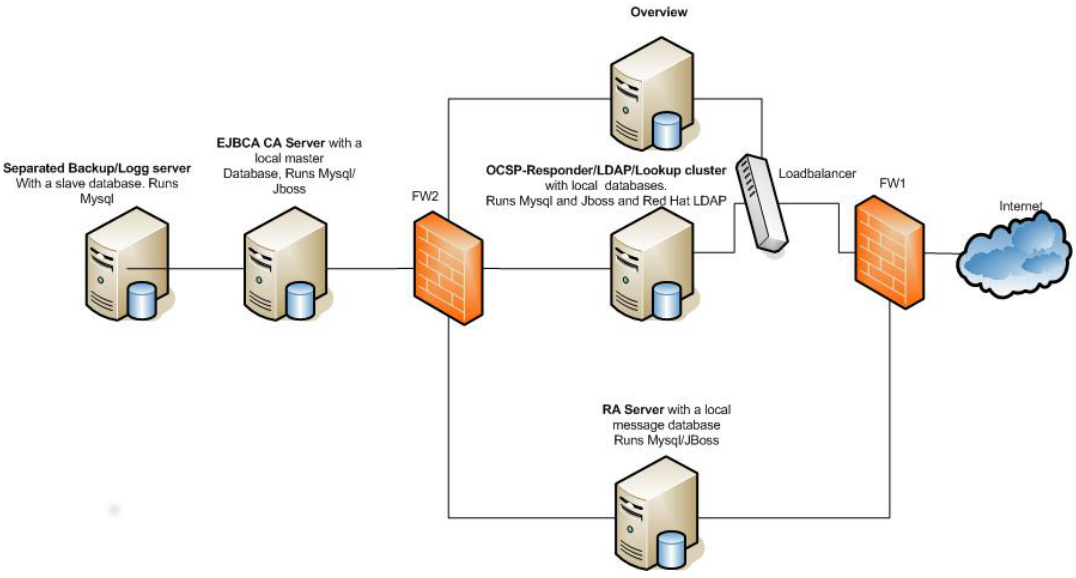


Abbildung 2.4: Komponenten des EJBCA-Architekturvorschlags, entnommen aus [EJB14c]

Die EJBCA ist eine Server-CA, die über Registration Authorities (RA) verwaltet wird. Das Rollenkonzept sieht die Rollen CA-Administrator und RA-Administrator vor. Der RA-Administrator kann über ein Webinterface einen Zertifikatsantrag einbringen. Der Kunde bekommt einen Benutzernamen und ein Passwort und kann sich sein Zertifikat ebenfalls über ein Webinterface abholen. Einstellungen bezüglich der Zertifikatsvorlagen und RA-Berechtigungen können nur von einem CA-Administrator durchgeführt werden.

Die CA-Software unterstützt Zertifikatsvorlagen mit Möglichkeiten für Fixwerte und variable Werte, die von der RA angepasst bzw. hinzugefügt werden können. Für die Konfiguration von individuellen Zertifikatserweiterungen, wie der Dienstleistereigenschaft, sind eine Konfiguration in einer Java-Properties-Datei und eine Neuübersetzung der Software notwendig. Die benötigten Konfigurationen für die Dienstleistereigenschaft sind in Anhang B beschrieben.

Die Erstellung aller Testzertifikate war möglich, einzig das Zertifikat „Zertifikat mit Qualified Certificate Statement“ unterscheidet sich von der Vorlage. In den Vorlagenzertifikaten wird das QC-Statement nach RFC3039 mit dem Identifier `pkixQCSyntax-v1` versehen. Die getestete Version der EJBCA unterstützt diesen Identifier nicht mehr, stattdessen wird die Version 2 (`pkixQCSyntax-v2`) nach RFC 3739 verwendet (s. Anhang B).

2.3.3 XCA – X Certificate and key management

Die XCA ist eine Open-Source-CA-Software und unter `http://xca.sourceforge.net` erreichbar. Beschrieben wird die Software von der zugehörigen Sourceforge-Homepage [Hoh13] wie folgt:

> „X Certificate and Key management is an interface for managing asymetric keys like RSA or DSA. It is intended as a small CA for creation and signing certificates. It uses the OpenSSL library for the cryptographic operations.“

Bei der XCA handelt es sich um ein Einzelplatzsystem, im Hintergrund wird die OpenSSL-Bibliothek verwendet.

Über die Benutzeroberfläche können die einfachsten Zertifikatseigenschaften statisch gesetzt bzw. können so auch Vorlagen für die spätere Verwendung angelegt werden. Das Erzeugen dynamischer Inhalte auf der Basis einer Kundendatenbank ist automatisch nicht möglich, jedoch werden alle Erweiterungen, die in einem PKCS#10 Request angefordert werden, auch in das Zertifikat aufgenommen. Für aufwendigere Zertifikatserweiterungen wird auf die OpenSSL-Konfiguration verwiesen. Dazu vermerkt das Handbuch unter [Hoh13]:

> „Advanced
> Any extension, not covered on the other tabs can be added here as defined in OpenSSL nconf. The validity can be checked by clicking Validate. All extensions from all tabs will be shown here to see them all in their final form. Click on Edit to continue editing the extensions here.“

Ein Rollenkonzept ist nicht vorgesehen; jeder, der ein Zertifikat ausstellen kann, hat Zugriff auf den privaten Schlüssel des Wurzelzertifikats. Dieser private Schlüssel ist

in einer Container-Datei abgelegt und mit einem Passwort geschützt. Es besteht die Möglichkeit, mehrere Container-Dateien anzulegen, um damit verschiedene Wurzelzertifikate zu trennen.

Die Testzertifikate konnten entsprechend den Vorlagen erstellt werden, jedoch muss für jedes Zertifikat die erweiterte Konfigurationsmöglichkeit verwendet werden. Dabei handelt es sich um OpenSSL-Konfigurationseinträge der in Listing 2.1 dargestellten Form:

```
1  authorityInfoAccess=@aia_sect
2  certificatePolicies=ia5org,@polsect
3  crlDistributionPoints=@crl_section
4
5  [polsect]
6  policyIdentifier=1.2.40.0.17.1.20
7  CPS.1="http://www.a-trust.at/docs/cp/a-sign-ssl"
8
9  [aia_sect]
10 caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-ssl-03.crt
11 OCSP;URI.1=http://ocsp.a-trust.at/OCSP
12
13 [crl_section]
14 URI.1=ldap://ldap.a-trust.at/out=a-sign-SSL-03,o=A-Trust,c=AT?
       certificaterevocationlist?base?objectclass=eidCertificateAuthority
```

Listing 2.1: XCA-Zertifikate mit mehreren Domainnamen im Subject Alternative Name

Der `authorityInfoAccess` muss in den erweiterten Funktionen gesetzt werden, da über das GUI nur ein Eintrag möglich ist. Die `certificatePolicies` können über das GUI nicht gesetzt werden. Der Eintrag für `crlDistributionPoints` ist zwar auch über das GUI möglich, jedoch können dort nur HTTP-URLs eingetragen werden bzw. gibt es Probleme beim Parsen der LDAP-URL.

Die benötigten Einstellungen für die Erstellung der Testzertifikate sind in Anhang C aufgelistet. Die Testzertifikate und Einstellungen sind auf der beigelegten CD-ROM enthalten (s. Anhang H).

Beim Erstellen einer Zertifikatswiderrufsliste mit der CA-Software ist es möglich, das Gültig-bis-Datum vor dem Gültig-ab-Datum zu wählen. Die gleichen Datumseinstellungen beim Generieren eines Zertifikats führen zu einer Warnmeldung, können jedoch akzeptiert werden. Diese Einstellung ist im zugehörigen Standard nicht explizit untersagt, jedoch ist die Verwendung nicht sinnvoll. Die zutreffenden Passagen aus [DC08] lauten:

> „The field is represented as a SEQUENCE of two dates: the date on which the certificate validity period begins (notBefore) and the date on which the certificate validity period ends (notAfter)[...]. The validity period for a certificate is the period of time from notBefore through notAfter, inclusive.“

2.3.4 SimpleAuthority

SimpleAuthority ist eine CA-Software, die sowohl als freie Version, Kaufversion und als Enterprise-Version verfügbar ist. Der zugehörige Webauftritt ist unter `http://simpleauthority.com` erreichbar. Die Software wird auf der zugehörigen Webseite [Sim14] folgendermaßen beschrieben:

> „SimpleAuthority is a fully functional Certification Authority, or Certificate Authority (CA), that is designed to be very easy to use. It generates and manages keys and certificates that provide cryptographic digital identities for people and/or computer servers. [...] Unlike most CA products, SimpleAuthority does not require specialist PKI knowledge or supporting components like an external database. It is built on The Legion of the Bouncy Castle cryptographic library."

SimpleAuthority kann als Einzelplatzssystem („free version") oder als Kommandozeilen-Script bzw. Web-Interface (beides nur als „enterprise license") ausgeführt werden. In der frei verfügbaren Version können nur Zertifikate entsprechend der vier vorgefertigten Templates erstellt werden. Für die Anpassungen dieser Vorlagen ist eine „standard license" notwendig. Als Kryptographiebibliothek wird Bouncycastle[2] verwendet.

Im Auslieferungszustand sind vier Zertifikatsvorlagen enthalten. Interessant für die Erstellung der Testzertifikate sind jedoch nur die „SSL Server" und die „General Purpose" Vorlagen. Diese erlauben es dem Benutzer, einfache Zertifikatseigenschaften zu setzen; erweiterte Einstellungen und benutzerdefinierte Zertifikatserweiterungen können nicht konfiguriert werden. Es fehlt z.B. die Erweiterung `authorityInfoAccess`. Diese gibt an, wo das übergeordnete Zertifikat abrufbar ist und wohin OCSP-Anfragen gesendet werden können. Zudem sind die Elemente für das Zertifikat-Subject eingeschränkt. Daher ist es nicht möglich, die Felder `givenName` und `surname` für Vorname und Nachname hinzuzufügen. Ein weiteres Problem zeigt sich beim Hinzufügen der Zertifikatserweiterung `crlDistributionPoints`: Die in der Vorlage eingetragenen URL wird zwar in das Zertifikat aufgenommen, jedoch werden automatisch der Name des CA-Zertifikats und die Dateierweiterung `.crl` angehängt. Bei Verwendung einer LDAP-URL für diese Zertifikatserweiterung entsteht das in Listing 2.2 gezeigte Ergebnis:

```
1  SEQUENCE {
2    OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
3    OCTET STRING, encapsulates {
4      SEQUENCE {
5        SEQUENCE {
6          [0] {
7            [0] {
8              [6]
9            'ldap://ldap.a-trust.at/ou=a-sign-SSL-03,o=A-Trus'
10           't,c=AT?certificaterevocationlist?base?objectclas'
11           's=eidCertificationAuthority/Root-Ca-01.crl'
```

[2] http://bouncycastle.org

```
12              }
13            }
14          }
15        }
16      }
17    }
```

Listing 2.2: simpleAuthority-Zertifikate mit falschem CRL Distribution Point

Aufgrund dieser Einschränkungen war es nicht möglich, alle Zertifikate nach den Vorlagen zu erstellen. Es kann ein Zertifikat nach der Vorlage „Zertifikat mit mehreren Domainnamen im Subject Alternative Name“ erstellt werden, jedoch fehlt, wie schon beschrieben, der `authorityInfoAccess`. Die Zertifikate mit Dienstleistereigenschaft A.4 und QC-Statement A.3 können mangels der Möglichkeit, die jeweilige Zertifikatserweiterung einzufügen, nicht erstellt werden, ebenso wenig das Zertifikat mit UPN A.2, da die Vorlagen für den `subjectAltName` keinen UPN zulassen.

Ähnlich der XCA ist kein Rollenkonzept vorgesehen. Das Passwort des privaten Schüssels des CA-Zertifikats wird beim ersten Anwenden von der Software abgefragt. Anschließend können beliebige Einstellungen durchgeführt bzw. Zertifikate ausgestellt werden.

2.3.5 OpenSSL-CA

OpenSSL ist eine Open-Source-Implementierung von Kryptographiealgorithmen und Protokollen. Zu den vielen Bestandteilen zählt auch eine CA-Software, die über Kommandozeilenbefehle gesteuert wird. Der offizielle Webauftritt von OpenSSL ist unter `https://www.openssl.org` erreichbar. Für die Analyse werden vorkompilierte Pakete von `http://slproweb.com/products/Win32OpenSSL.html` verwendet. Nachfolgend wird die Beschreibung von OpenSSL von der zugehörigen Homepage [Ope14b] wiedergegeben.

> „The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.
>
> OpenSSL is based on the excellent SSLeay library developed by Eric A. Young and Tim J. Hudson. The OpenSSL toolkit is licensed under an Apache-style licence, which basically means that you are free to get and use it for commercial and non-commercial purposes subject to some simple license conditions.“

In OpenSSL werden die Einstellungen für das Benutzerzertifikat über Extension-Dateien getätigt. Aus diesen Dateien, dem Zertifikat-Request und dem CA-Zertifikat ergeben sich die Felder des Benutzerzertifikats. Die Möglichkeiten für benutzerdefinierte Erweiterungen im Zertifikat sind nicht beschränkt, das heißt, es können beliebige Strukturen für Erweiterungen erstellt werden. Einschränkungen gibt es jedoch beim Zertifikat-Subjekt, das über die Kommandozeilen-Programme nur mit einem Teil der erlaubten Werte befüllt werden kann. So ist es zum Beispiel nicht möglich, die Felder `title` und `serialNumber` aufzunehmen. Diese Einschränkungen gelten jedoch nur für die mitgelieferten Programme. Die Bibliothek selbst kann beliebige Werte im Zertifikat-Subject anlegen, wie bereits in 2.3.3 gezeigt wurde.

Ein Rollenkonzept ist für die Kommandozeilen-Programme nicht vorgesehen. Der private Schlüssel der CA wird mit einem Passwort geschützt, und jeder, der die zugehörigen Dateien und dieses Passwort hat, kann Zertifikate ausstellen.

Abgesehen von den Einschränkungen im Subject war es möglich, alle Testzertifikate zu erstellen. Der Ablauf der Ausstellung und die OpenSSL-Extension-Dateien sind im Anhang D aufgelistet.

2.3.6 Microsoft-CA

Die Microsoft Windows Certificate Authority ist Teil der Windows-Server-Reihe und kann in zwei verschiedenen Versionen betrieben werden, als „Standalone CA“ oder als „Enterprise CA“. Die „Standalone“ Version wird ohne Windows Domain oder Active Directory betrieben und ist im Umfang auf Standardzertifikate ohne Möglichkeiten der Anpassung eingeschränkt. Alle Informationen, die in das Zertifikat aufgenommen werden, müssen im Zertifikat-Request angegeben sein. In der Microsoft-Dokumentation [Mic13] wird dies wie folgt beschrieben:

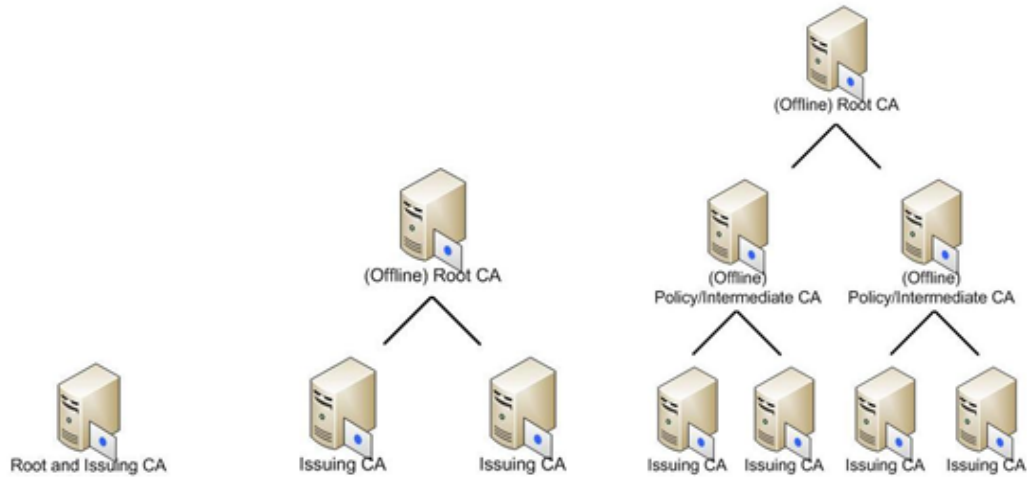
> „Stand-alone CAs do not require Active Directory and do not use certificate templates. If you use stand-alone CAs, all information about the requested certificate type must be included in the certificate request.“

Die „Enterprise“ Version benötigt eine Windows Domain und ein Active Directory, damit Zertifikate ausgestellt und verwaltet werden können. Diese Version bietet die Verwendung von Zertifikatsvorlagen, die einige Werte des Zertifikat-Request überschreiben bzw. hinzufügen können. Die Beschreibung aus der Microsoft-Dokumentation [Mic13] ist wie folgt:

> „Enterprise CAs are integrated with Active Directory. They publish certificates and CRLs to Active Directory. Enterprise CAs use information stored in Active Directory, including user accounts and security groups, to approve or deny certificate requests. Enterprise CAs use certificate templates. When

> a certificate is issued, the enterprise CA uses information in the certificate template to generate a certificate with the appropriate attributes for that certificate type.“

Von der Microsoft-CA kann ein OCSP-Server-Dienst als eigenständige Komponente abgetrennt werden. Zur besseren Ausfallsicherheit können sowohl CA-Dienst als auch OCSP-Dienst in einem Microsoft-Cluster betrieben werden. Als Designvorschlag werden von Microsoft die in Abbildung 2.5 dargestellten Strukturen empfohlen (vgl. [Pyl09]). Je nach Größe des Unternehmens und örtlicher Verteilung der „Issuing CA“Instanzen ist eine „Single/One Tier Hierarchy“ für kleine Unternehmen gedacht, hingegen eine „Two Tier Hierarchy“ oder „Three Tier Hierarchy“ für größere Unternehmen und/oder räumlich getrennte Unternehmenssitze.



(a) Single/One Tier Hierarchy (b) Two Tier Hierarchy (c) Three Tier Hierarchy

Abbildung 2.5: Microsoft-Hierarchie-Empfehlungen [Pyl09]

Als Grundlage für die weitere Analyse und weitere Tests wird die Microsoft-CA der Windows Server 2012 Datacenter Evaluaton Edition[3] in der Version „Enterprise“ verwendet.

Der auffälligste Unterschied der Microsoft-CA zu den bisher getesteten Programmen ist, dass die Eigenschaften und Erweiterungen des Zertifikats hauptsächlich vom Benutzer festgelegt werden und von der CA-Software erweitert, teilweise übersteuert werden können.

Die Erstellung der Testzertifikate war in zwei Fällen nicht möglich. Beim Erstellen sowohl nach der Vorlage „Zertifikate mit Dienstleistereigenschaft“ als auch nach der Vorlage „Zertifikate mit Qualified Certificate Statement“ können die Zertifikatserweiterungen nicht eingetragen werden. Weitere Probleme zeigen sich bei dem Feld `serialNumber`

[3] http://technet.microsoft.com/en-us/evalcenter/hh670538.aspx

im Subject, dieses kann nicht angelegt werden. Auffällig ist auch, dass in jedes Zertifikat automatische Microsoft-Zertifikatserweiterungen aufgenommen werden. Unter anderem wird der Inhalt des Feldes `extKeyUsage` zusätzlich als `Application Policies` eingetragen.

Das Rollenkonzept der Microsoft-CA bietet auf der Basis der Zertifikatsvorlage die Möglichkeit, die Benutzerzertifikate automatisch zu erstellen oder die Ausstellung erst nach einer vorherigen Bestätigung von beliebig vielen Administratoren durchzuführen. Der private Schlüssel des Wurzelzertifikats wird von der CA-Software verwaltet und nach der Autorisierung durch das Rollenkonzept ohne Passworteingabe angewendet.

2.3.7 OpenCA PKI / OpenXPKI

OpenCA PKI ist eine Open-Source-Lösung der „OpenCA Labs - opensource security and identify management solutions“. Der zugehörige Webauftritt ist unter `https://pki.openca.org/` erreichbar. Die Beschreibung der CA-Software lautet wie folgt:

> „The OpenCA PKI Project is a collaborative effort to develop a robust, full-featured and Open Source out-of-the-box Certification Authority implementing the most used protocols with full-strength cryptography world-wide. OpenCA is based on many Open-Source Projects. Among the required software there are OpenLDAP, OpenSSL, Apache Project, Apache mod_ssl.
>
> The project development is divided in two main tasks: studying and refining the security scheme that guarantees the best model to be used in a CA and developing software to easily setup and manage a Certification Authority.“

OpenXPKI bzw. OpenX509PKI ist eine CA-Software, die sich im Oktober 2005 von der OpenCA abgespalten hat und seit November 2009 als „feature-complete“ anzusehen ist. Jedoch wurde bisher (Stand Mai 2014) kein offizielles Release ausgegeben. Der Webauftritt der OpenXPKI ist unter `http://www2.openxpki.org/` erreichbar.

Die OpenCA PKI kann auf mehrere Komponenten aufgeteilt bzw. aus mehreren Komponenten kombiniert werden. Als Beispiel einer Aufteilung in einem weltweiten Unternehmen wird im „OpenCA Guide“ (s. [Gro05]) Abbildung 2.6 angegeben. Obwohl in der Abbildung und der Dokumentation sechs Komponenten aufgezeigt sind, werden für den Vergleich mit anderen analysierten CA-Systemen nur drei Systeme betrachtet. Diese sind das CA-Interface für die Erstellung von Zertifikaten bzw. Sperrinformationen, ein OCSP-Server und der RA-Server, entsprechend in der Abbildung die Punkte Node, Public und RA.

Ähnlich dem Konzept der EJBCA (Kapitel 2.3.2) wird auch die OpenCA über Registration Authorities (RA) verwaltet. Der Kunde bestellt über eine öffentliche Webseite ein Zer-

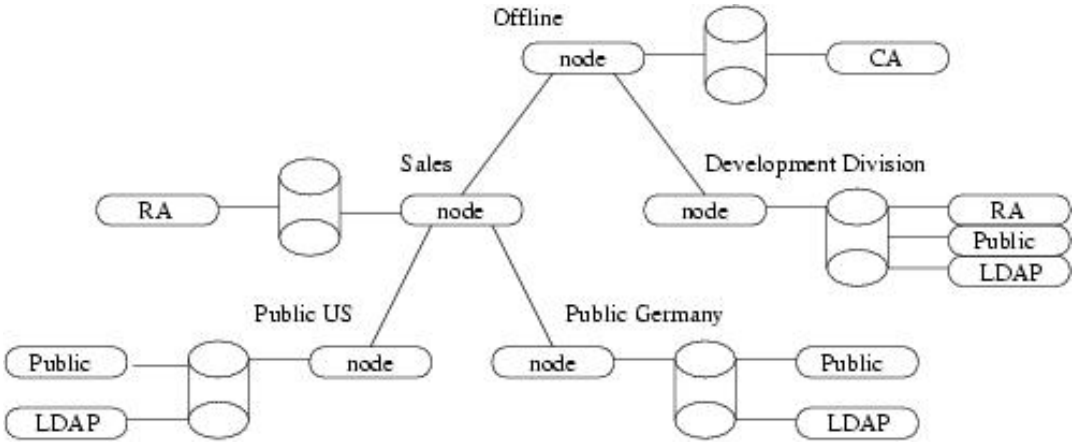


Abbildung 2.6: Beispiel OpenCA-PKI-Aufbau [Gro05]

tifikat, anschließend muss er mit einem Ausweis in einer RA authentifiziert werden. Der RA-Administrator genehmigt („approved“) den Zertifikatsantrag. Abschließend muss ein CA-Administrator das Zertifikat ausstellen, und der Kunde erhält eine E-Mail mit einer Kennzahl, um mit dieser das ausgestellte Zertifikat herunterzuladen. Dadurch sind im Rollenkonzept der OpenCA mehrere CA-Administratoren und Registration Authorities mit jeweils mehreren RA-Administratoren vorgesehen.

Die Zertifikatsvorlagen bestehen aus OpenSSL-Konfigurationsdateien, wie sie schon bei der OpenSSL-CA verwendet wurden. Daher können für die Testzertifikate entsprechend dieselben Einstellungen mit kleineren Anpassungen verwendet werden (s. OpenSSL-Einstellungen Anhang D). Die tatsächlichen Konfigurationsdateien für die OpenCA sind auf der beigelegten CA im Ordner `/analyse/openca/` enthalten (s. Anhang H).

Die Erstellung der Testzertifikate ist auf der Basis der OpenSSL-Konfigurationen möglich, jedoch gibt es Einschränkungen für die verwendeten Felder im Zertifikat-Subject. Die Felder `givenName` und `title` können über das Interface der OpenCA nicht eingetragen werden.

2.3.8 Symantec Managed PKI Certificate Service

Die Firma Symantec (`www.symantec.com`) bietet unter dem Namen „Symantec Managed PKI Services Test Drive“[4] eine Testversion ihrer Managed PKI Services an. Da es sich hierbei um ein bei Symantec gehostetes Server-System handelt, kann keine Aussage zu den Serverkomponenten bzw. den Möglichkeiten der Lastverteilung getroffen werden.

Das System bietet zwölf vorgefertigte Zertifikatsprofile (Zertifikatsvorlagen), die ange-

[4] https://testdrive-pki-account.symauth.com/account-manager/test-drive/new.xhtml

passt werden können. Ein Hinzufügen von Zertifikatserweiterungen ist nicht möglich, jedoch werden alle Felder für das Zertifikats-Subject unterstützt. Zusätzlich bietet das Management-Interface die Möglichkeit, Eingabedaten für die Zertifikatsprofile aus unterschiedlichen Quellen zu importieren, unter anderem von Microsoft-Active-Directory-Strukturen oder LDAP-Servern.

Das Rollenkonzept der Managed PKI sieht einen Administrator vor, der Zertifikate bestätigen und Konfigurationsänderungen durchführen kann. Benutzer können selbständig ein Zertifikat anfordern, jedoch muss der Anforderung vom Administrator zugestimmt werden.

Aufgrund der Einschränkungen der Managed PKI ist es nur möglich, das Zertifikat nach der Vorlage „Zertifikat mit UPN im Subject Alternative Name“ zu erstellen. Es ist nicht möglich, ein SSL-Server-Zertifikat zu erstellen; dies betrifft die Vorlagen „Zertifikat mit mehreren Domainnamen im Subject Alternative Name“ und „Zertifikat mit Dienstleistereigenschaft“. Da keine Einstellungen vorhanden sind, um benutzerdefinierte Zertifikatserweiterungen in die Zertifikatsprofile aufzunehmen, ist es nicht möglich, die benötigten Erweiterungen für die Vorlagen „Zertifikat mit Qualified Certificate Statement“ und „Zertifikat mit Dienstleistereigenschaft“ zu erstellen.

2.3.9 Weitere CA-Software

pyCA ist eine Open-Source-Software von Michael Ströder und unter `http://www.pyca.de/` verfügbar. Sie wird seit November 2003 nicht mehr aktiv weiterentwickelt. Im Prinzip handelt es sich bei dem Produkt um mehrere Webserver, die eine graphische Oberfläche für die OpenSSL-Bibliothek darstellen (s. [Str03a]). Die Software ist als Diplomarbeit von Michael Ströber an der Universität Karlsruhe entstanden (s. [Str99]). Das System besteht aus zwei Komponenten einem „Private CA System“ und einem „Public Server System“. Die meisten Konfigurationseinstellungen für die auszustellenden Zertifikate werden den OpenSSL-Konfigurationsdateien entnommen (s. [Str03b]). Im Hinblick auf die Ausstellung der Testzertifikate ergeben sich dieselben Einstellungen wie für die OpenSSL-Kommandozeilen-Version (vgl. Kapitel 2.3.5).

gnoMint ist eine Open-Source-Software von David Marín und unter `http://gnomint.sourceforge.net/` verfügbar. Zum letzten Mal wurde sie im August 2010 verändert und scheint seither nicht mehr aktiv entwickelt zu werden. Die Software erlaubt die Erstellung einfacher Zertifikate, die benötigten Zertifikatsdaten werden einer Zertifikatsrequest (PKCS#10) entnommen. Es gibt jedoch keine Möglichkeit, Erweiterungen in das Zertifikat einzufügen, weshalb eine Erstellung von Testzertifikaten nicht möglich ist.

TinyCa ist eine Open-Source-Benutzeroberfläche für OpenSSL, der zugehörige Webauftritt findet sich unter `http://www.sm-zone.net/`[5]. Die Software bietet die einfache Erstellung von Server- und Client-Zertifikaten, jedoch können keine Erweiterungen hinzugefügt werden, so dass auch hier die Erstellung der Testzertifikate nicht möglich ist.

PHP-CA PHP-CA (phpbasedca) ist eine Open-Source-Software und wurde im November 2006 gestartet; die letzte Änderung am Quellcode wurde Ende Februar 2007 durchgeführt. Das Projekt ist unter `http://php-ca.sourceforge.net/` erreichbar. Eine Installation der Software ist nicht möglich, da die Datenbankscripte fehlen. Aus der Beschreibung der Webseite und den Quellcode-Dateien lässt sich schließen, dass die PHP-CA OpenSSL verwendet, um Zertifikate auszustellen, so dass Zertifikatserweiterungen über OpenSSL-Zertifikatsvorlagen möglich sind.

r509 Ruby Certificate Authority ist eine Open-Source-Software von Paul Kehrer, Sean Schulte und Mike Ryan und steht unter `http://r509.org/` zur Verfügung. Sie basiert auf OpenSSL und hat das Ziel, eine vollständige Implementierung von RFC 3280/5280 zu sein. Zertifikatstemplates können über yaml-Dateien erstellt werden, sind aber in ihrer Funktionalität eingeschränkt. Bei der Erstellung der Testzertifikate konnte kein UPN eingetragen werden, die Dienstleistereigenschaft und das QC-Statement in der Konfigurationsdatei wurde ignoriert. Darüber hinaus konnten keine LDAP-URLs für den `crlDistributionPoint` angegeben werden, die Werte für den `subjAltName` sind auf fünf Eingaben beschränkt und die Felder `givenName`, `surname`, `title` und `serialNumber` im Subject können nicht hinzugefügt werden. Die Konfigurationen für die Testzertifikate sind in Anhang E aufgelistet.

2.3.10 Zusammenfassung

Eine Übersicht über die Testergebnisse ist in Tabelle 2.1 dargestellt. Bei der Analyse hat sich die starke Verbreitung von OpenSSL als Kryptographiebibliothek unter den CA-Software-Systemen gezeigt. Einige dieser Systeme bezeichnen sich selbst als graphische Oberfläche zur einfachen Handhabung der OpenSSL-Bibliothek und verwenden für erweiterte Einstellungen die von der Kryptographiebibliothek zur Verfügung gestellten Konfigurationsdateien.

[5] Webseite nicht mehr verfügbar, Link Web Archive: http://web.archive.org/web/20130906123542/http://www.sm-zone.net/

Wie in den Analyseergebnissen zu erkennen, ist die Unterstützung des Zertifikat-Subject eines der größten Probleme bei den getesteten Produkten. Die in den Testzertifikaten verwendeten Werte für dieses Feld sind im RFC 5280 (vgl. [DC08]) definiert. In Kapitel 4.1.2.6 werden für das Subject folgende „Implementation Requirements“ definiert:

> „The subject field is defined as the X.501 type Name. Implementation requirements for this field are those defined for the issuer field (Section 4.1.2.4).“

Das referenzierte Kapitel 4.1.2.4 beschreibt die Felder folgendermaßen:

> „Standard sets of attributes have been defined in the X.500 series of specifications [X.520]. Implementations of this specification MUST be prepared to receive the following standard attribute types in issuer and subject (Section 4.1.2.6) names:
>
> - country,
> - organization,
> - organizational unit,
> - distinguished name qualifier,
> - state or province name,
> - common name (e.g., “Susan Housley“), and
> - serial number.
>
> In addition, implementations of this specification SHOULD be prepared to receive the following standard attribute types in issuer and subject names:
>
> - locality,
> - title,
> - surname,
> - given name,
> - initials,
> - pseudonym, and
> - generation qualifier (e.g., “Jr.“, “3rd“, or “IV“).“

Bei der Erstellung der Testzertifikate gab es Probleme mit dem Feld `serialNumber` aus dem ersten Bereich, der als „MUST“ (muss, Requirement, absolutes Muss, vgl. [Bra07]) definiert ist. Problematisch waren auch die Felder `title`, `surname` und `givenName`, die im zweiten Bereich als „SHOULD“ (empfohlen, vgl. [Bra07]) definiert sind.

Schwierigkeiten zeigen sich bei der Unterstützung der untersuchten Zertifikatserweiterungen. Einträge im `subjectAltName`, wie in der Vorlage „Zertifikat mit mehreren Domainnamen im Subject Alternative Name“, sind durch eine breitere Unterstützung meist ohne größere Anpassungen zu erstellen. Für seltener verwendete Erweiterungen wie

das QC-Statement muss in vielen Fällen ein Eingriff in die erweiterten Konfigurationsdateien getätigt werden. Vor allem trifft dieser Umstand auf die Dienstleister-Erweiterung zu, die von keiner Standardsoftware unterstützt wird. Durch den verbreiteten Einsatz von OpenSSL werden zum Konfigurieren solcher Erweiterungen in den meisten Fällen auch die von OpenSSL unterstützten Konfigurationsdateien verwendet.

Ein Rollenkonzept wird bei den getesteten Produkten nur in vier Fällen unterstützt. Grundsätzlich wird zwischen der Rolle der Antragsprüfung und der Rolle der Zertifikatsausstellung unterschieden, wobei die zweite Rolle auch automatisierbar ist. Das Konzept der Rollenaufteilung und Berechtigungen unterscheidet sich nicht bzw. nur minimal von dem Konzept der derzeit eingesetzten Smarttrust CA.

Auffällig ist, dass in vielen der getesteten CA-Software-Systemen eine Installation mit einem CA-Zertifikat gleichzusetzen ist. Es besteht bei diesen Produkten nicht die Möglichkeit, mehrere CA-Schlüssel bzw. CA-Zertifikate gleichzeitig in einer Installation zu verwalten – bei der Generierung eines weiteren CA-Zertifikats wird das bestehende Zertifikat ersetzt.

Positiv anzumerken ist, dass fast alle getesteten Produkte aktuelle Hash- und Kryptographiealgorithmen unterstützen.

Name	Kategorie	Zertifikatsvorlagen	Rollenkonzept	Zertifikat-Subject	SAN[1]	UPN[2]	QC Statement[3]	Dienstleister[4]	ECDSA	SHA 256
Smarttrust CA	Server-System	✓	✓	✓	✓	✓	✓	✓	✓	✓[5]
EJBCA PKI CA	Server-System	✓	✓	✓	✓	✓	✓	✓	✓	✓
XCA	Einzelplatzsystem	✓		✓	✓	✓	✓	✓	✓	✓
SimpleAuthority	Einzelplatzsystem Kommandozeilen-Script Server-System	✓			✓				✓	
OpenSSL-CA	Kommandozeilen-Script	✓			✓	✓	✓	✓	✓	✓
Microsoft-CA	Server-System	✓	✓		✓	✓			✓	✓
OpenCA PKI	Server-System	✓	✓		✓	✓	✓	✓	✓	✓
Symantec Managed PKI Services	Server-System	✓	✓	✓		✓			✓	✓
pyCA	Server-System	✓			✓	✓	✓	✓	✓	✓
gnoMint	Einzelplatzsystem Kommandozeilen-Script			✓					✓	✓
TinyCa	Einzelplatzsystem								✓	✓
PHP-CA	Server-System								✓	✓
r509	Server-System	✓			✓				✓	✓

[1] Webserverzertifikat mit mehreren Domains im Subject Alternative Name (SAN)
[2] Zertifikat mit UPN
[3] Zertifikat mit Qualified Certificate Statement (QC)
[4] Webserverzertifikat mit Dienstleistereigenschaft(OID)
[5] In der aktuellsten Version der Software wird SHA256 untertützt

Tabelle 2.1: Zusammenfassung der Analyse von CA-Software-Systemen

3 Präzisierung der Aufgabenstellung

Wie in der Einleitung bereits beschrieben, soll eine CA-Server-Software entwickelt werden, die das bestehende System ablöst. Als Vorgabe werden das bestehende vorgelagerte System und die bestehenden Systemschnittstellen angesehen. Durch diese ergibt sich das Systemumfeld, das in Abbildung 3.1 dargestellt ist.

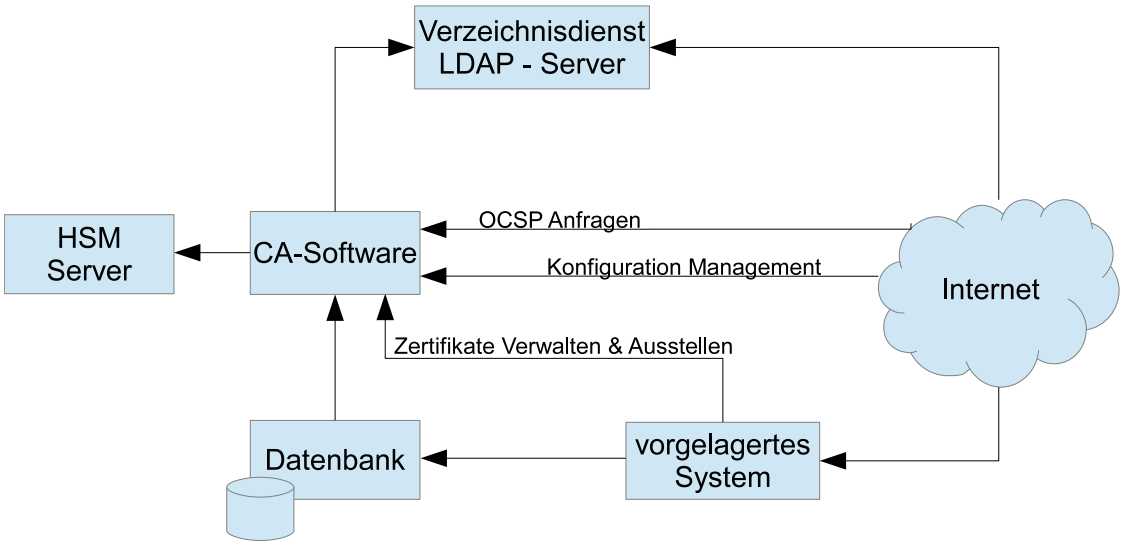


Abbildung 3.1: Systemumfeld der CA-Software

3.1 Anforderungen an die Lösung

Das CA-Server-System wird auf autonomen Servern in einem Rechenzentrum betrieben und muss daher ohne Benutzerinteraktion arbeiten. Konkret ergeben sich dadurch folgende Anforderungen:

- Die Erstellung von Sperrinformationen soll automatisch erfolgen.
- Nach Bestätigung durch eine berechtigte Person (RO) soll ein Benutzerzertifikat automatisch ausgestellt werden.
- OCSP-Anfragen werden automatisch beantwortet.
- Anpassungen der Konfiguration werden durch eine eigene Management-Schnittstelle durchgeführt, ohne dass eine Person physischen Zugriff auf die Server hat.

Das CA-Server-System muss in das bestehende vorgelagerte System integriert werden. Die zur Verfügung gestellten Schnittstellen sollen dabei nicht verändert werden. Als Basis für die weitere Beschreibung der CA-Software-Lösung dienen die Abläufe innerhalb der CA-Software, welche bereits in [Hag13, Kapitel 3.1] bearbeitet wurden.

Anforderungen an die Konfigurationsänderungen

Konfigurationsänderungen müssen von einer oder zwei berechtigten Personen (SO) bestätigt werden. Eine Auflistung der derzeit zur Verfügung stehenden Konfigurationsoptionen und der Anzahl der Personen, die die jeweilige Aktion bestätigen müssen, ist in Anhang F zu finden. Diese Auflistung dient als Basis für die Implementierung der Management-Schnittstelle.

Anforderungen an die Verfügbarkeit

Die Verfügbarkeit der einzelnen Teilaufgaben der CA-Software wurde bereits in [Hag13, Kapitel 2.5] genauer erläutert. Das Ergebnis der Untersuchung ist in Tabelle 3.1 zusammengefasst.

Aufgabe	**Verfügbarkeit (Std. x Tage)**
Benutzerzertifikat, Widerruf und Sperre	Arbeitszeit: innerhalb von 3 Stunden sonst: innerhalb von 6 Stunden
Benutzerzertifikat, Ausstellung	10x5
Verwaltung von CA-Zertifikat, Ausstellen, Widerruf und Sperre	8x5
Konfigurationsänderungen	8x5
CRL-Erstellung und -Veröffentlichung	24x7
Statusabfragen über OCSP	24x7
Statusabfragen A-Trust Verzeichnisdienst	24x7

Tabelle 3.1: Verfügbarkeit einzelner Teilaufgaben

Wie in der Tabelle 3.1 zu erkennen ist, sind die drei wichtigsten Aufgaben die CRL-Erstellung und -Veröffentlichung, die OCSP Abfragen und der Verzeichnisdienst (LDAP). Diese Dienste sowie deren Verfügbarkeit und erlaubte Stehzeiten sind im österreichischen Signaturgesetzt reglementiert (siehe [Sig08, §10 Absatz 5]). Die Verfügbarkeiten für den Widerruf und die Sperre von Benutzerzertifikaten ergibt sich ebenfalls aus dem österreichischen Signaturgesetzt (siehe [Sig08, §10 Absatz 4]).

Die Verwaltung von CA-Zertifikaten bzw. Konfigurationsänderungen werden nur zu den Büroöffnungszeiten durchgeführt, daher ergeben sich die angeführten Zeiten für acht Stunden pro Tag bei fünf Tagen die Woche. Die Verfügbarkeit der Ausstellung von Benutzerzertifikaten wird durch verschiedene Kundenprojekte und Vertragspartner bestimmt. Aus den gesammelten Vereinbarungen ergeben sich zehn Stunden pro Tag an fünf Tagen die Woche.

3.2 Design- und Implementierungsprobleme

Durch die aktuelle Problemstellung und die Anforderungen und aufgrund der Erfahrung mit dem derzeit eingesetzten CA-Server-System und der Analysen aus Kapitel 2.3 ergeben sich folgende Probleme, die das Design und die Implementierung eines CA-Server-Systems betreffen.

Komponentenaufteilung

Die Aufteilung der Aufgaben der CA-Software auf einzelne Komponenten ist ein essentieller Punkt beim Design des Gesamtsystems. Bei dieser Entscheidung ist vor allem die Verfügbarkeit der Einzelaufgaben (s. Kapitel 3.1) zu berücksichtigen.

Ein Beispiel für die Berücksichtigung der Verfügbarkeit bei der Komponentenaufteilung ist die Erstellung der Sperrlisten: Eine hohe Last bei Anfragen zum Ausstellen von Zertifikaten darf nicht dazu führen, dass Sperrlisteninformationen (CRL) zu spät oder gar nicht erzeugt und publiziert werden.

Auch sollten aus dem Internet erreichbare Schnittstellen so abgesichert sein, dass durch Angriffe (z.B. Denial-of-Service-Attacken) keine bzw. nur eine geringe Beeinflussung der restlichen Komponenten stattfindet.

Zusätzlich ist zu berücksichtigen, dass die privaten Schlüssel der CA-Zertifikate in Hardware-Security-Modulen gespeichert werden müssen. Diese Module verhindern, dass die privaten Schlüssel nicht bzw. von keiner nicht autorisierten Person exportiert werden können.

Ausfallsszenarien/Lastverteilung

Derzeit wird die CA-Software in einem Microsoft-Cluster (Hot-Cold) betrieben; eine zusätzliche Backup-Instanz ist an einem anderen Standort eingerichtet. Dieses Konzept ermöglicht es, bei einem Problemfall sofort auf ein zur Verfügung stehendes Backup-System umzuschalten. Jedoch werden alle Anfragen nur an das aktive System gesendet. Es sind Alternativen zu diskutieren, die auch eine Lastverteilung ermöglichen, und deren Auswirkungen auf die Implementierung der CA-Software zu erwägen. Dabei sind vor allem Synchronisierungsprobleme, Datenbankzugriffe, fortlaufende Seriennummern und eine doppelte Erstellung der Sperrlisten zu berücksichtigen.

Zertifikatsformate und -regeln

Die Unterscheidung zwischen Formaten und Regeln für den Aufbau von Zertifikaten besteht in den Modifikationsmöglichkeiten. Formate beinhalten im allgemeinen Fixwer-

te welche für jedes Zertifikat des entsprechenden Typs gleichbleibend sind. Dies betrifft z.B. Zertifikatserweiterungen wie `certificatePolicies`, `crlDistributionPoints`, `keyUsage` und `authorityInfoAccess`. Die Zertifikatsregeln beschreiben jene Erweiterungen, welche trotz selben Zertifikatstyps unterschiedlichen Inhalt haben. Dazu gehören z.B. die Inhalte des `subject`, `subjectAltName` und der UPN für die Chipkartenanmeldung. Oftmals wird durch diese Regeln die Möglichkeit geboten Zertifikatsfelder zu verschieben, zu löschen oder aus weiteren Datenquellen zu laden.

Die derzeitigen Zertifikatsformate und -regeln werden in Konfigurationsdateien gespeichert. Dies führt zu mehreren Problemen. Das erste Problem ergibt sich bei der Modifikation der Konfigurationsdateien. Diese werden vom CA-System nicht protokolliert und können zu einem späteren Zeitpunkt so nicht nachvollzogen werden. Für ein Sicherheitssystem ist es jedoch wichtig, feststellen zu können, welche Person zu welchem Zeitpunkt Änderungen durchgeführt hat.

Ein weiteres Problem ist die Synchronisierung der Konfigurationsdateien über mehrere Instanzen der CA-Software. Die CA-Software ist redundant ausgelegt, die Einstellungen müssen somit auf mindestens drei Servern durchgeführt werden. Daher ist eine Lösung mit automatischer Synchronisierung erstrebenswert.

Vor allem bei der Analyse der CA-Software-Systeme zeigen sich Probleme bei der Implementierung der Zertifikatsregeln. Die möglichen Felder für das Zertifikat-Subject sind den aktuellen Standards zu entnehmen und müssen in der CA-Software verwaltet werden können. Bei den Testzertifikaten mit QC-Statement oder Dienstleistereigenschaft zeigt sich, dass die Vorlagen bzw. Regeln auch derzeit noch nicht bekannte Zertifikatserweiterungen unterstützen müssen.

Zu beachten ist, dass ca. 100 Zertifikatsformate aus der SmartTrust-CA in das neue System übernommen werden müssen. Daher wäre eine möglichst ähnliche Lösung erstrebenswert.

Sperrlisteninformationen (CRL)

Wie in Kapitel 3.1 beschrieben, ist die rechtzeitige Erstellung der Sperrinformationen eine der wichtigsten Aufgaben. In einer CRL sind zwei Zeiten enthalten. Die Gültig-ab-Zeit gibt den Zeitpunkt der Erstellung der Sperrliste an, die Gültig-bis-Zeit den Zeitpunkt, ab dem die Sperrliste nicht mehr gültig ist. Zusätzlich zu der zeitgerechten Generierung der nächsten Sperrlisten ist diese auch im sogenannten „immediate-issue“-Verfahren (Sofortausstellung bei Widerruf oder Sperre) zu erstellen. Dadurch ergeben sich einige Spezialfälle bei der Berechnung der Gültig-ab- und der Gültig-bis-Daten der CRL.

Zusätzliche Anforderungen an die Gültig-bis-Zeit sind im RFC 5280 enthalten (s. [DC08, Kapitel 5.1.2.5]):

> „This field indicates the date by which the next CRL will be issued. The next CRL could be issued before the indicated date, but it will not be issued any later than the indicated date. CRL issuers SHOULD issue CRLs with a nextUpdate time equal to or later than all previous CRLs.“

Daraus ergibt sich die Anforderung, dass das Gültig-bis-Datum gleich oder später als das Gültig-bis-Datum aller bisher ausgestellten CRLs sein muss.

Bei der Berechnung der Gültigkeit der Sperrlisten werden in der CA-Software zwei Zeiten angegeben, ein Update-Intervall und eine Margin-Zeit. Das Update-Intervall gibt die Zeit zwischen zwei CRL-Erstellungen an. Die Margin-Zeit ist die Zeit, um die das Update-Intervall erhöht wird, um sicherzustellen, dass immer eine gültige CRL verfügbar ist (vgl. [S.d07, S. 83]). Gleichzeitig geben Update-Intervall und Margin-Zeit addiert die maximale Zeitspanne zwischen dem Gültig-bis-Wert und dem Gültig-ab-Wert an.

Ein weiteres Problem sind die Größe der Sperrliste und die damit verbundenen Komplikationen bei der Signatur, der Veröffentlichung und der Verteilung.

OCSP-Server

Der OCSP-Server wurde bereits vor einigen Jahren durch eine Eigenentwicklung ersetzt, jedoch ergeben sich notwendige Anpassungen durch aktuelle Anforderungen und neue Standards, wie z.B. den RFC 5019[6] „The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments“ und den RFC 6960[7] „X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP“ aus dem Jahr 2013. In diesen Standards wurden die Möglichkeiten der Signatur der OCSP-Antwort überarbeitet. Dies betrifft unter anderem die Frage, welches Zertifikat die OCSP-Antwort unterschreiben darf bzw. soll. Der aktuelle OCSP-Server verwendet eine Signatur-Variante, von der in den neuen Standards dringend abgeraten wird.

3.3 Ausschluss nicht relevanter Themen

Die nachfolgenden Themen werden aus den genannten Gründen aus der aktuellen Betrachtung ausgeschlossen.

Rollenkonzept/Registrierungsstellenkonzept

Durch die abzulösende CA-Software und das vorgelagerte System ist bereits ein Rollenkonzept mit Registration Authorities vorgegeben. Es ist den Konzepten in den Kapiteln

[6] http://tools.ietf.org/html/rfc5019
[7] http://tools.ietf.org/html/rfc6960

2.3.2 und 2.3.7 sehr ähnlich. Die Erkenntnisse der Analyse bringen in diesem Fall keinen Mehrwert über das bestehende System. Daher wird eine Veränderung des Rollenkonzeptes im weiteren Verlauf der Arbeit nicht betrachtet.

Attributszertifikate

Obwohl die derzeit eingesetzte CA-Software Attributszertifikate erstellen könnte, wird dieses Feature derzeit nicht verwendet. Zudem sind keine Projekte mit Attributszertifikaten geplant. Daher werden bei der aktuellen Untersuchung und der Eigenentwicklung der CA-Software Attributszertifikate nicht berücksichtigt.

Cross-Zertifizierungen

Bei einer Cross-Zertifizierung werden die CA-Zertifikate zweier Zertifizierungsdiensteanbieter vom jeweils anderen zusätzlich als Cross-Zertifikat ausgestellt. Dadurch wird das Vertrauen bzw. die Gleichstellung dieser beiden CA-Zertifikate PKI-technisch ausgedrückt. Aus technischer Sicht handelt es sich bei dieser Prozedur um die Erstellung eines CA-Zertifikats, wobei der private Schlüssel nicht selbst kontrolliert wird. Derzeit werden keine Cross-Zertifizierungen ausgestellt, ein solcher Fall ist daher bei der weiteren Untersuchung nicht zu berücksichtigen.

4 Systemkonzept

4.1 Komponentenaufteilung

Wie schon in den Anforderungen an die Komponentenaufteilung beschrieben, darf durch die einzelnen Teilaufgaben bzw. Komponenten keine Beeinflussung anderer Komponenten stattfinden. Dies gilt vor allem für jene Aufgaben, die in Kapitel 3.1 mit einer Verfügbarkeit von 24x7 gekennzeichnet sind.

Als Basis für dieses Kapitel dient die bereits durchgeführte Analyse in [Hag13, Kapitel 4.1]. Die dort bearbeiteten Varianten werden nachfolgend kurz beschrieben und um neue Erkenntnisse erweitert. Zusätzlich werden einige neue Möglichkeiten der Komponentenaufteilung dargestellt. Grundlage der neuen Erkenntnisse bildet die Analyse aus Kapitel 2.3.

4.1.1 Monolithischer Ansatz

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.1.1] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Für die weitere Vorgehensweise wird im ersten Beispiel der einfachste Ansatz angenommen. Wie in Abbildung 4.1 dargestellt, werden alle Aufgaben in einer Komponente zusammengefasst. Anhand dieses Modells wird in den nachfolgenden Unterkapiteln die weitere Aufteilung besprochen.

Vorteile:

- Es ist keine aufwendige Interprozesskommunikation zwischen den einzelnen Komponenten notwendig.

Nachteile:

- Ein Programmfehler in einer Teilaufgabe kann im Extremfall den Ausfall des Gesamtsystems bedeuten.
- Hohe Last oder DoS-Angriffe[8] an der Kommunikationsschnittstellen können zu einer negativen Beeinflussung der restlichen Teilaufgaben führen.

Der in den Anforderungen beschriebene Fall der Beeinflussung der Sperrlistenerstellung durch hohe Last bei der Zertifikatsausstellung wird in dieser Variante nicht behoben. Ebenso wird eine Absicherung der Kommunikationsschnittstelle bzw. eine Trennung

[8] Denial-of-Service-Angriff

zu den restlichen Komponenten gewünscht, welche mit dieser Variante nicht gegeben ist.

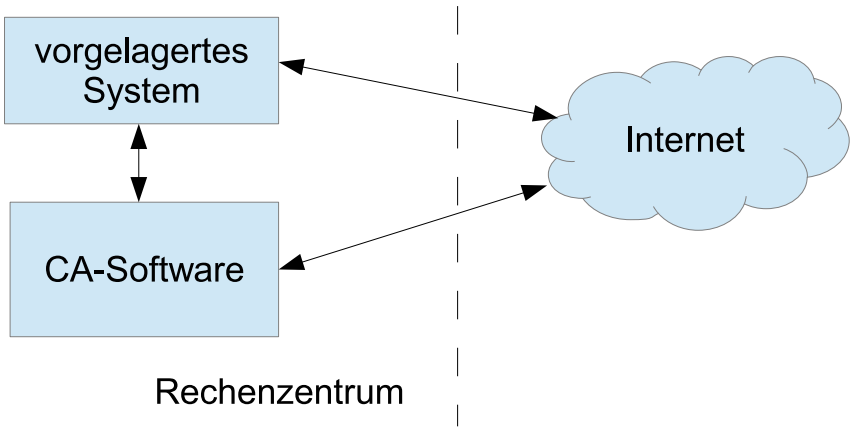


Abbildung 4.1: Komponenten des monolithischen Aufbaus

4.1.2 Kommunikationsschnittstelle als eigene Komponente

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.1.2] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Der auffälligste Nachteil der vorherigen Variante ist die Integration der Kommunikationsschnittstelle mit den restlichen Funktionen. Daher beschreibt dieser Ansatz die Trennung in Kommunikationsserver und CA-Software. Dies ist in Abbildung 4.2 dargestellt.

Vorteile:

- Ungültige und fehlerhafte Kommunikationsversuche werden durch den Kommunikationsserver aussortiert.
- Anfragen die eine reine Konfigurationstätigkeit ausführen, werden direkt in dem Kommunikationsserver durchgeführt.
- Verringerte Last bei der CA-Software Komponente.

Nachteile:

- Anfragen werden an zwei Stellen auf ihre Gültigkeit geprüft, im Kommunikationsserver und in der CA-Software.

In der bisherigen Analyse wurde jedoch noch nicht berücksichtigt, dass auch eine hohe Last durch gültige Requests erzeugt werden kann. Durch erhöhte gleichzeitige Zertifikatsausstellungen oder Widerrufe kann es trotz des Kommunikationsservers zu einer großen Anzahl an Anfragen an die CA-Software kommen, welche die Ausstellung der Sperrinformationen beeinflussen können.

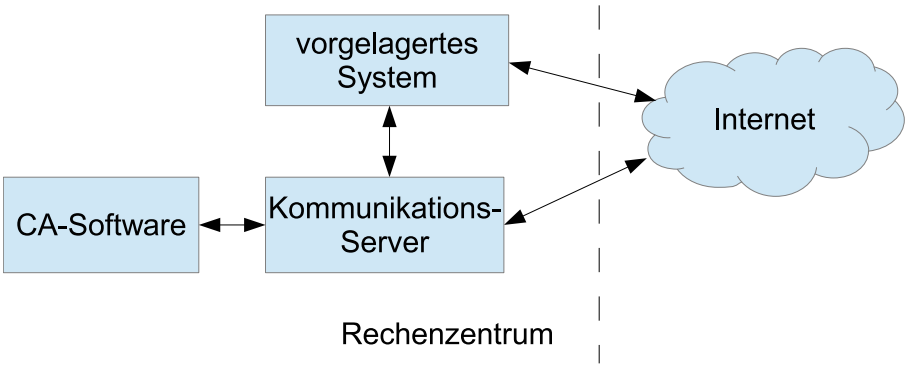


Abbildung 4.2: Kernkomponente mit Kommunikationsserver

4.1.3 Architekturvorschlag der Software EJBCA

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.1.3] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Die EJBCA wurde bereits in der Analyse in Kapitel 2.3.2 vorgestellt, der zugehörige Architekturvorschlag ist in Kapitel 2.3.2 in Abbildung 2.4 dargestellt.

Vorteile:

- Trennung der aus dem Internet direkt erreichbaren Server und der Backend-Server.
- Durch diese Asynchronität der Zertifikatsausstellung können Anfragen seriell abgearbeitet werden. Dies führt zu geringeren Wechselwirkungen zwischen den einzelnen Teilaufgaben.

Nachteile:

- Asynchrone Verarbeitung (Pollingverfahren) der eingehenden Anfragen der Zertifikatsausstellung und des OCSP-Server. Dadurch sind hohen Latenzzeiten bei den Anfragen zu erwarten.
- Anfragen werden an zwei Stellen auf ihre Gültigkeit geprüft, in der empfangenden Komponenten und im Backend-System.

4.1.4 Architektur der SmartTrust-CA

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.1.4] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Die SmartTrust-CA ist die derzeit eingesetzte und abzulösende CA-Software, die entsprechende Architektur ist in Abbildung 4.3 dargestellt.

Vorteile:

- Wartezeiten und Wiederholzyklen bei der Veröffentlichung haben keine Auswirkung auf die anderen Aufgabengebiete bzw. Komponenten.

Nachteile:

- Zeitversetzung der Veröffentlichung.
- Erfolgreiche Zertifikatsausstellung oder erfolgreicher Zertifikatswiderruf bedeuten nicht, dass die entsprechenden Informationen bereits veröffentlicht wurden.

Zu den bereits aufgelisteten Punkten ist noch zu erwähnen, dass die Bedenken hinsichtlich einer Wechselwirkung der einzelnen Teilaufgaben in einer Komponente weiterhin zutreffen.

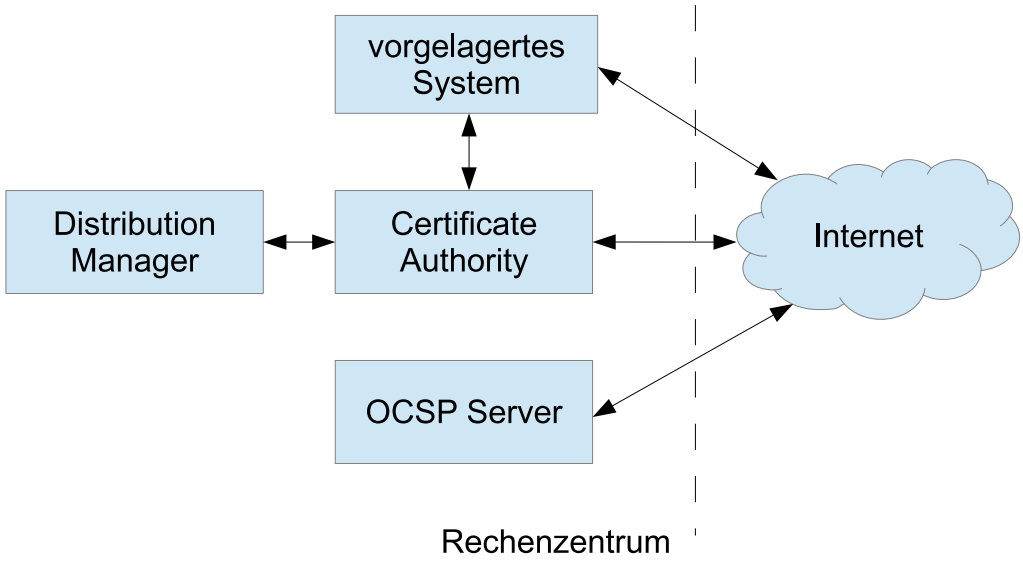


Abbildung 4.3: Komponenten der SmartTrust-CA

4.1.5 Architektur der Microsoft-CA

Die von Microsoft vorgeschlagenen Architekturen wurden in der Analyse in Kapitel 2.3.6 in Abbildung 2.5 dargestellt. Empfohlen wird eine „Two Tier Hierarchy“ mit einer Offline-Root-CA. Zur Erhöhung der Ausfallsicherheit wird ein Cluster-Betrieb empfohlen, der im Dokument [CBK10] beschrieben ist. Dieser Cluster-Betrieb unterstützt nur die CA-Rollen, während die OCSP-Rolle und die optionale Rolle Certificate Enrollment auf einem eigenen Server installiert werden müssen (vgl. [CBK10, S. 8]). Daraus ergibt sich die in Abbildung 4.4 dargestellte Komponentenaufteilung.

Der Vorteil der Trennung des OCSP-Servers von den restlichen Aufgaben besteht darin, dass sich hochfrequentierte OCSP-Anfragen nicht auf die Zertifikatsausstellung und

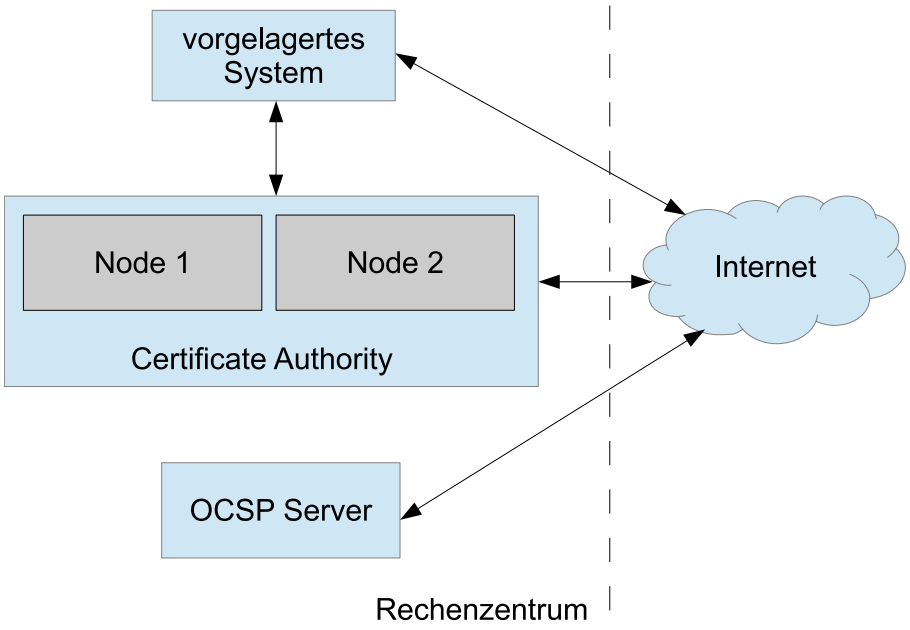


Abbildung 4.4: Komponenten der Microsoft-CA

die Sperrlistenerstellung auswirken. Jedoch sind die Aufgaben der Erstellung der Sperrlisteninformation und der Erstellung von Zertifikaten weiterhin in einer Komponente vereint. Damit ergeben sich die bereits in den vorherigen Kapiteln beschriebenen Nachteile der gegenseitigen Beeinflussung.

4.1.6 Komponentenaufteilung OpenCA/OpenXPKI

Im „OpenCA Guide“ wird ein Beispiel einer Komponentenaufteilung angegeben, das in Abbildung 2.6 in Kapitel 2.3.7 dargestellt ist. Grundlegend teilt sich die OpenCA PKI in die Komponenten CA-Service, Public Interface, RA-Interface, Node-Management und OCSP-Server. Vereinfacht dargestellt ergibt sich die Abbildung 4.5. Bei dieser Abbildung ist jedoch zu beachten, dass sich die Aufgabengebiete des vorgelagerten Systems mit jenen des RA-Interface und des Public-Interface überschneiden.

Wie schon bei der Komponentenaufteilung in Variante „Kommunikationsschnittstelle als eigene Komponente“ (vgl. Kapitel 4.1.2) liegt der Vorteil dieser Aufteilung darin, dass der CA-Server nicht von außen erreichbar ist. Daher haben DoS-Attacken keine Auswirkung auf die Erstellung der Sperrlisteninformationen. Durch die Aufteilung in mehrere Komponenten, die auch von außen erreichbar sind, entstehen mehr Angriffsmöglichkeiten bzw. Komponenten, die gegen Angriffe abgesichert werden müssen. Obwohl der OCSP-Server als eigene Komponente extrahiert wurde und der CA-Server von außen nicht erreichbar ist, besteht weiterhin das in Kapitel 4.1.2 beschriebene Problem (hohe Last durch gültige Requests).

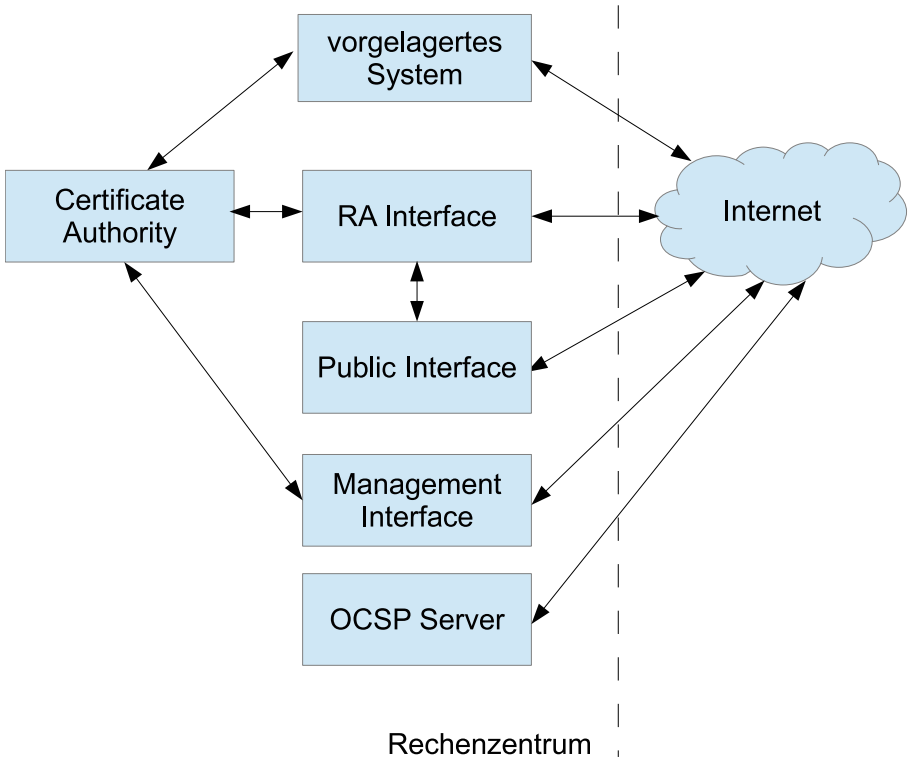


Abbildung 4.5: Komponenten der OpenCA/OpenXPKI

4.1.7 Komponentenaufteilung unter Berücksichtigung der Verfügbarkeit

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.1.5] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Diese Variante verfolgt den Ansatz der Aufteilung der Komponenten nach der Verfügbarkeits-Tabelle (s. Kapitel 3.1 Tabelle 3.1). Dadurch ergeben sich die folgenden Komponenten:

- CA-Server
- CRL-Server
- Kommunikationsserver
- Publication-Server
- OCSP-Server

Das Zusammenspiel dieser Komponenten ist in Abbildung 4.6 dargestellt.

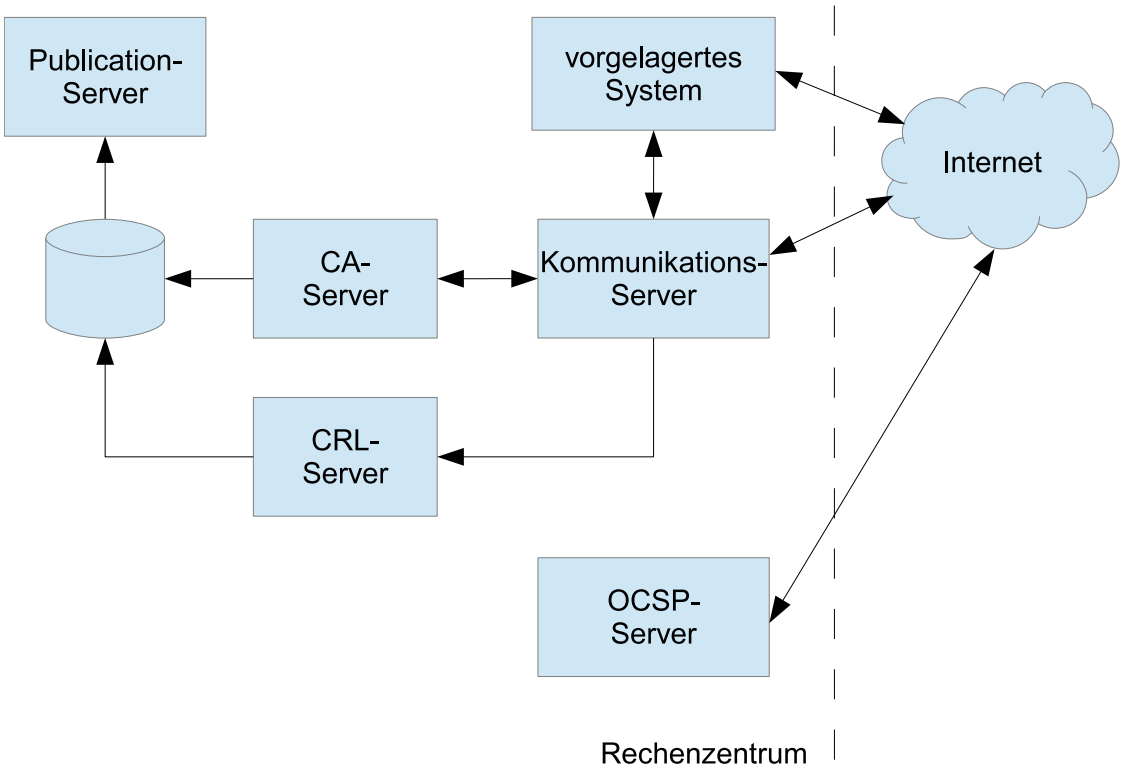


Abbildung 4.6: Komponenten - Verfügbarkeit

Vorteile:

- Unterschiedliche Priorisierung der einzelnen Komponenten.
- Hohe Last und DoS-Attacken werden bereits im Kommunikationsserver abgefangen.

Nachteile:

- Erhöhte interne Kommunikation zwischen den einzelnen Komponenten.

In den bisherigen Betrachtungen wurde nicht berücksichtigt, dass eine hohe Last auch durch gültige Anfragen entstehen kann, welche zu einer Überlastung der Komponenten Kommunikationsserver und CA-Server führen. Durch die Aufteilung des CA-Servers und CRL-Servers beeinflussen sich diese Komponenten nicht gegenseitig. In diesem Fall muss der Publication-Server jedoch eine Priorisierung der Publication-Anfragen vornehmen; die Veröffentlichung der Sperrlisten muss höher priorisiert sein als jene der Zertifikate. Dies ist durch die Kommunikation über Datenbankeinträge einfach umzusetzen.

4.1.8 Autonome dezentrale Systeme (ADS)

Ein interessanter Ansatz ist jene, in [CGC11] beschriebene, Architektur. Die Problematik der hohen Verfügbarkeit, kurzer Responsezeiten und gegenseitiger Beeinflussung von Teilaufgaben wird durch die Aufteilung der CA-Server in mehrere eigenständige Noden gelöst. Wie in der Arbeit beschrieben wird eine eingehende Anfrage mit einem Content Code (CC) versehen, welcher eindeutig für die jeweilige Anfrage ist (s. [CGC11, Seite 8, Abs. 1]). Die so präparierte Nachricht wird an das ADS übermittelt. Auf Basis des eingefügten Content Code wählt eine Node die Anfrage aus und bearbeitet diese. Eine solche Node beinhaltet zwei Komponenten, zum einen den empfangenden und sendenden Dienst (z.B. Kommunikations Server), zum anderen einen CA-Server. Eine schematische Darstellung des Gesamtsystems ist in Abbildung 4.7 zu sehen.

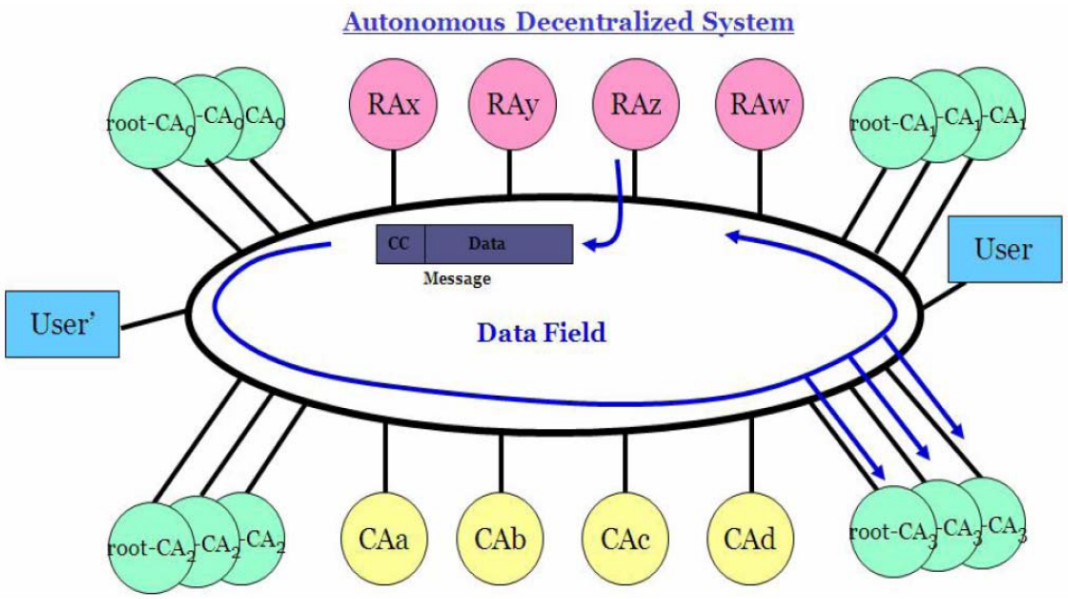


Abbildung 4.7: Schematische Übersicht der ADS-PKI [CGC11, Figure 5]

Die Objekte RAx bis RAw beschreiben Registration Authorities, welche die Anfrage zur Zertifikatsausstellung an das Data Field (DF) übergeben. Durch den Content Code in der Nachricht (Message) wird die Anfrage von nur einer zuständigen CA (CAa bis CAd bzw. root-CA0 bis root-CA3) verarbeitet.

Durch eine Gleichverteilung der Anfragen auf die zuständigen Noden ist eine parallele Verarbeitung bzw. ein Loadbalancing möglich. Eine Erweiterung des ADS durch zusätzliche Noden ist jederzeit möglich, dadurch werden sowohl die parallele Verarbeitung, das Loadbalancing als auch die Ausfallssicherheit erhöht.

Eine Problematik die in [CGC11] nicht beschrieben wird, ist die Erstellung von Sperrlisten. Durch die Gleichberechtigung einzelner Node des selben Aufgabengebietes (in Abbildung 4.7 z.B. die 3 Noden der root-CA0) kommt es zu dem Problem, dass diese gleichzeitig die Sperrlisten erstellen. Daher ist eine Synchronisation dieser Noden

erforderlich, so dass die Sperrlistenerstellung nicht mehrfach ausgeführt wird.

4.1.9 Zusammenfassung

Eine tabellarische Übersicht der Komponentenaufteilung ist in Tabelle 4.1 dargestellt. Hier ist zu beachten, dass eine Trennung der Veröffentlichung in eine eigene Komponente auch bedeutet, dass die Veröffentlichung zeitversetzt durchgeführt wird. Durch den Vorteil, dass die Zeiten der Veröffentlichung (Wiederholzeiten und Timeouts) nicht in die Zeit der Erstellung von Zertifikat/Sperrlisteninformation einfließen, ergibt sich auch der Nachteil der zeitversetzten Veröffentlichung.

Die Trennung des OCSP-Servers in eine eigene Komponente ist eine Variante, die von allen analysierten CA-Server-Systemen unterstützt wird. In diesen Systemen gibt es nur geringe Interaktionen des OCSP-Servers mit den restlichen Komponenten. Er kann daher ohne großen Aufwand eigenständig betrieben werden.

Ziel der Komponentenaufteilung ist auch der Schutz der zentralen Komponenten. Daher ist es sinnvoll, die CA-Services zu schützen und einen Zugriff von außen nicht zu erlauben. Durch die Einführung eines Kommunikationsservers werden die CA-Services geschützt, und DoS-Attacken und eine hohe Last beeinflussen die Zertifikatserstellung nicht.

Die Erstellung der Sperrlisteninformationen bei den analysierten CA-Server-Systemen unterscheidet sich von den Anforderungen, wie sie in Kapitel 3.1 und Kapitel 3.2 beschrieben wurden. Gefordert wird vom CA-System die rechtzeitige Erstellung von Sperrlisten in einem Abstand von z.B. zwei Stunden. Angesichts der Anforderungen an das zu entwickelnde CA-Server-System ist die Aufteilung der CA- und CRL-Services in getrennte Komponenten eine sinnvolle Variante, um die Stabilität und Verfügbarkeit zu steigern. Diese Entscheidung wirkt sich zudem auf die Abtrennung einer Komponente aus, welche die Veröffentlichung der Zertifikate und Sperrlisteninformationen behandelt.

Name	**Schutz vor DoS-Attacken**	**Schutz vor hoher Last an Kommunikationsschnittstelle**	**synchrone interne Kommunikation**	**keine oder geringe interne Kommunikation**	**keine doppelte Requestprüfungen**	**Trennung Veröffentlichung** (eigene Komponente)	**Trennung OCSP-Server** (eigene Komponente)	**CA-Services geschützt** (nicht von außen erreichbar)	**Trennung CA- und CRL-Service** (jeweils eigenständige Komponenten)
Monolithischer Ansatz			✓	✓	✓				
Kommunikationsserver als eigene Komponente	✓		✓					✓	
Architekturvorschlag der EJBCA	✓	✓					✓	✓	
SmartTrust-CA			✓		✓	✓	✓		
Microsoft-CA			✓	✓	✓		✓		
OpenCA/OpenXPKI			✓				✓	✓	
Komponentenaufteilung unter Berücksichtigung der Verfügbarkeit	✓	✓	✓			✓	✓	✓	✓
Autonome dezentrale Systeme (ADS)	✓	✓			✓	n.a.	n.a.	✓	

n.a. . . . nicht angegeben/anwendbar

Tabelle 4.1: Zusammenfassung der Komponentenaufteilung

4.2 Ausfallszenarien und Lastverteilung

Ziel dieses Kapitels ist die Diskussion mögliche Lösungen zur Erhöhung der Ausfallsicherheit des CA-Server-Systems sowie der Lastverteilung. Zwei Möglichkeiten wurden bereits in der Analyse bzw. der Komponentenaufteilung erwähnt; diese und weitere werden nachfolgend angeführt und miteinander verglichen.

4.2.1 Failover Cluster

Bei einem Failover Cluster werden mehrere unabhängige Server (Node) zu einem virtuellen Server zusammengeschlossen. Ein Service bzw. eine Komponente ist nur auf einer Node aktiv. Tritt jedoch auf dieser Node ein Fehler auf, wird der Service bzw. die Komponente auf einer anderen Node aktiviert. Dies ist in Abbildung 4.8 graphisch dargestellt.

Die Beschreibung aus [Mic09, S. 7] ist wie folgt:

> „A failover cluster is a group of independent computers, or nodes, that are physically connected by a local-area network (LAN) or a wide-area network (WAN) and that are programmatically connected by cluster software. The group of nodes is managed as a single system and shares a common namespace. The group usually includes multiple network connections and data storage connected to the nodes via storage area networks (SANs). The failover cluster operates by moving resources between nodes to provide service if system components fail.
>
> Normally, if a server that is running a particular application crashes, the application will be unavailable until the server is fixed. Failover clustering addresses this situation by detecting hardware or software faults and immediately restarting the application on another node without requiring administrative intervention – a process known as failover. Users can continue to access the service and may be completely unaware that it is now being provided from a different server (see Figure 1 [Abbildung 4.8; Anmerk. d. Verf.])."

Diese Technologie hat den Vorteil, dass ein fehlerhafter Service bzw. eine fehlerhafte Hardware automatisch behoben wird, indem eine bereitstehende Node aktiviert wird. Dies geschieht in den meisten Fällen automatisch, der Kunde bemerkt die Veränderung im System nicht.

Die Komponenten aus Kapitel 4.1 sind jeweils als eigenständige Services zu implementieren und im Cluster als Ressource zu konfigurieren. Durch die Abhängigkeiten der Komponenten, z.B. zwischen Kommunikationsserver und CA-Server, ist es notwen-

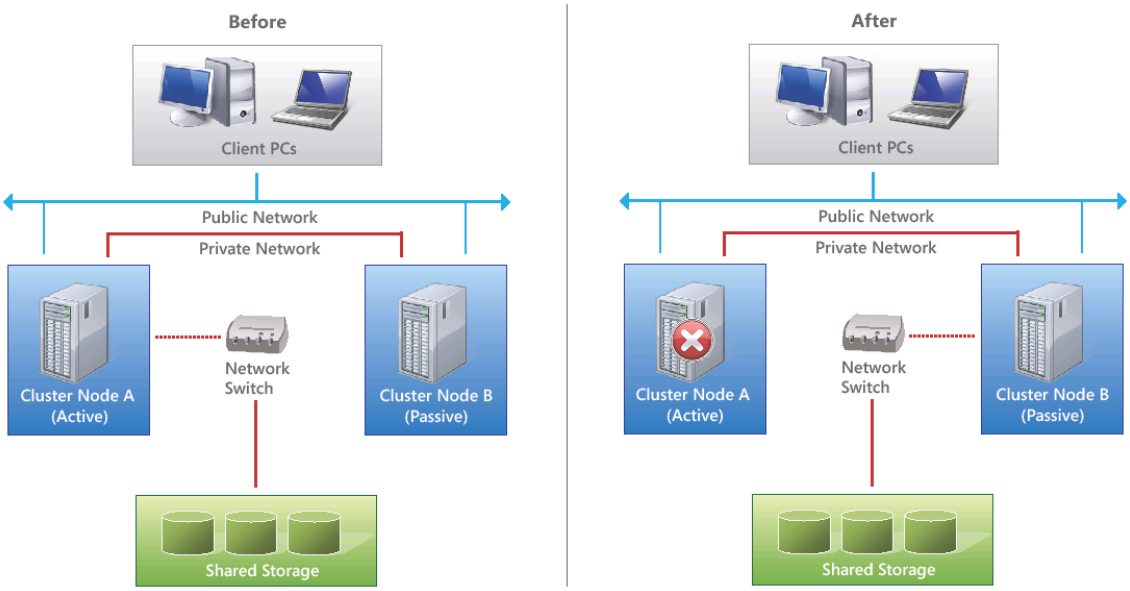


Abbildung 4.8: Microsoft-Cluster, entnommen [Mic09, S. 8]

dig, diese auch in den Cluster-Services zu definieren. Daraus folgt, dass dann, wenn ein Fehler in einem Service bzw. einer Komponente erkannt wird, auch alle zugehörigen Services bzw. Komponenten abgeschaltet und auf einer anderen Node aktiviert werden. Bei einer großen Anzahl an Abhängigkeiten kann sich dadurch die Zeit des Umschaltens von einer Node zur anderen erheblich erhöhen.

Failover Cluster ermöglichen geplante Wartungszeiten, ohne die Verfügbarkeit des Gesamtsystems zu beeinflussen. Auf den inaktiven Nodes des Clusters können Wartungsarbeiten durchgeführt werden, während die aktiven Nodes normal arbeiten.

Ein weiterer Vorteil gegenüber der Loadbalancing-Variante ist, dass in vielen Fällen keine Anpassung der Software notwendig ist und es zu keinen Problemen der Synchronisation zweier gleicher Services kommt.

4.2.2 Loadbalancing

Beim Loadbalancing werden die eingehenden Requests durch einen Algorithmus auf die unterschiedlichen Server aufgeteilt. Je nach Loadbalancing-Algorithmus (Round-Robin, Weighted Round-Robin, Least-Connection, lastabhängige Algorithmen) werden die Anfragen gleichmäßig über die verfügbaren Server verteilt unter Berücksichtigung gestörter bzw. fehlerhafter Server. Eine schematische Darstellung ist in Abbildung 4.9 dargestellt.

Die Vorteile dieser Variante ist die erhöhte Verfügbarkeit, kombiniert mit erhöhter Performance durch Verteilung der Anfragen. Wie auch bei der Failover-Cluster-Methode können Wartungsarbeiten auf inaktiven Nodes durchgeführt werden, ohne dadurch die

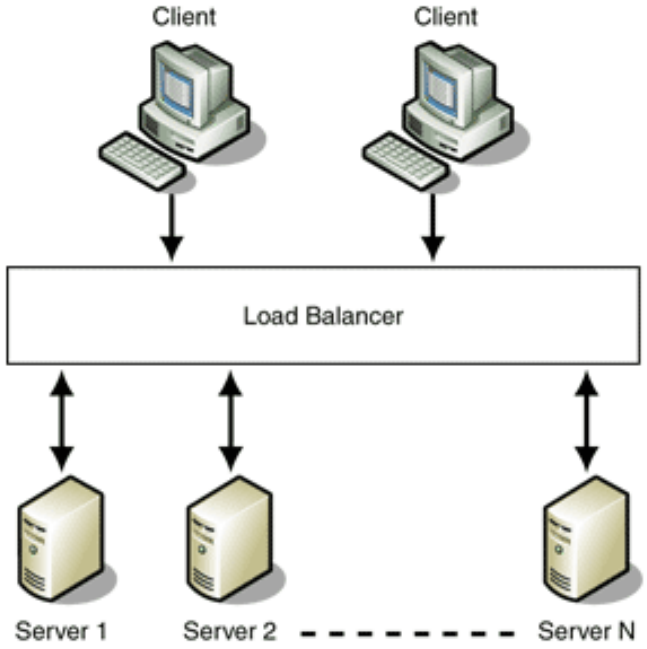


Abbildung 4.9: Loadbalancing Standardzustand, entnommen [Mic14b, Abschnitt: Load-Balanced Cluster]

Verfügbarkeit des Gesamtsystems zu beeinflussen.

Der größte Nachteil dieser Methode besteht in der Komplexität bei der Entwicklung der Komponenten. Da zur gleichen Zeit mehrere gleiche Komponenten aktiv und in Verwendung sind, müssen diese synchronisiert werden. Eine genauere Betrachtung anhand der Komponenten aus Kapitel 4.1.7 sieht folgendermaßen aus:

Kommunikationsserver: Der Kommunikationsserver behandelt die Anfragen von dem vorgelagerten System oder aus dem Internet. Bei der Installation auf mehreren Nodes müssen die aktuellen Client-Sessions zwischen den einzelnen Instanzen des Services synchronisiert werden. Der effizienteste Weg einer solchen Synchronisation ist über eine gemeinsame Datenbank. Zusätzlich müssen gleichzeitige Datenbankzugriffe serialisiert werden, dies wird über Datenbanktransaktionen realisiert.

CA-Server: Der CA-Server erstellt und verwaltet Zertifikate, auch hier ist eine Serialisierung der Datenbankzugriffe über Datenbanktransaktionen notwendig.

CRL-Server: Der CRL-Server erstellt zu vordefinierten Zeiten aktuelle Sperrlisteninformationen, denn eine Serialisierung mehrerer Komponenten ist in diesem Fall nicht ausreichend. Mit einer einfachen Serialisierung würden alle Server-Instanzen hintereinander die benötigte CRL ausstellen. Zur Lösung dieser Problematik können folgende Techniken eingesetzt werden:

Eine Server-Instanz wird als Master-Instanz gekennzeichnet. Bei zwei Nodes kann

die Kennzeichnung manuell erledigt werden, bei mehreren Nodes wird die Kennzeichnung üblicherweise durch ein Handshake beim Start des Services durchgeführt. Die Master-Instanz behandelt die regelmäßigen Sperrlisteninformationen, Sofortausstellungen von Sperrlisten werden von allen Instanzen durchgeführt. Die aktuellen Sperrlistenzeiten und Seriennummern werden über die Datenbank synchronisiert. Zur Erkennung eines Ausfalls der Master-Instanz müssen die anderen Instanzen regelmäßig die Verfügbarkeit der Master-Instanz abfragen.

Eine weitere Technik ist die Synchronisierung über Datenbankeinträge und Transaktionen. Hierbei muss von der CRL-Server-Instanz zuerst geprüft werden, ob die aktuelle CRL nicht bereits von einer anderen Instanz erstellt und in diesem Fall die CRL-Erstellung unterlassen wurde. Im Optimalfall werden unterschiedliche Sperrlisteninformationen ebenfalls parallel von mehreren Instanzen ausgestellt.

Publication-Server: Ebenso wie beim CA-Server muss der Datenbankzugriff über Transaktionen serialisiert werden.

OCSP-Server: Der OCSP-Server benötigt keine Serialisierung. Anfragen werden lokal bearbeitet und benötigen nur Lesezugriffe auf die Datenbank oder einen Verzeichnisdienst.

Die Erkennung von fehlerhaften Services wird in diesem Fall von einem Loadbalancing-Server durchgeführt. Dieser muss von außen entscheiden, ob der Service auf der jeweiligen Node fehlerhaft ist oder nicht. Im Gegensatz dazu hat ein Failover Cluster die Möglichkeit, auf lokale Informationen der Node zuzugreifen, um diese Entscheidung zu treffen.

4.2.3 CA-Hierarchie/Verteilung

Wie bereits in der Analyse in Kapitel 2.3 von einigen CA-Systemen empfohlen, kann die CA-Struktur in einer Hierarchie aufgebaut werden (vgl. Abbildung 2.5 und Abbildung 2.6 in Kapitel 2.3). In diesem Konzept wird für jede untergeordnete CA-Instanz ein eigener Server mit getrennten Datenbanken verwendet. Jede Node erstellt und verwaltet nur Zertifikate und Sperrlisteninformationen, welche der eigenen CA zugeordnet sind. Die verschiedenen Server benötigen nur eine geringe Kommunikation untereinander. Eine mögliche Hierarchie könnte wie in Abbildung 4.10 aussehen.

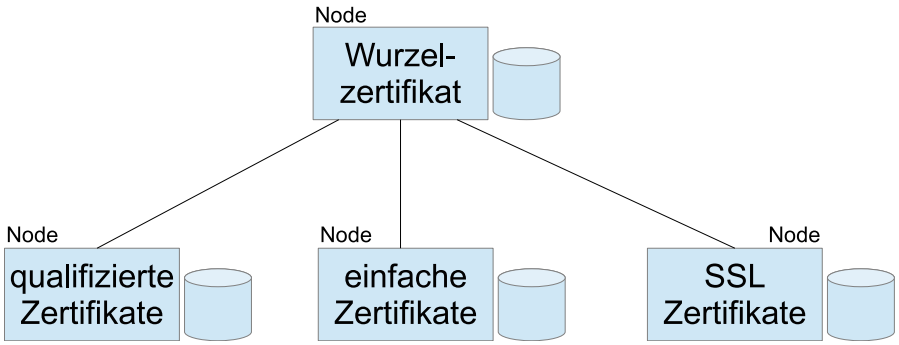


Abbildung 4.10: CA-Hierarchie/Verteilung

Im derzeitigen Konzept der Smarttrust-CA werden die Zertifikatsseriennummern über alle CA-Zertifikate global fortlaufend vergeben, diese Vorgehensweise würde bei einer CA-Hierarchie/Verteilung zu Problemen bei der Synchronisierung unter den verschiedenen Nodes führen. Im RFC5280 ist die Zertifikatsseriennummer wie folgt definiert (vgl. [DC08]):

> „The serial number MUST be a positive integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate). CAs MUST force the serial number to be a non-negative integer.“

Daraus lässt sich erkennen, dass für jede ausstellende CA eine eigene Nummerierung durchgeführt werden kann. Dadurch kann die Berechnung der Zertifikatsseriennummern pro Node erfolgen und muss nicht über die Nodes synchronisiert werden.

Bei einer Gleichverteilung der auszustellenden Zertifikate über die Nodes würde diese Lösung eine Lastverteilung wie in Kapitel 4.2.2 zur Loadbalacing bedeuten, jedoch ohne die Komplexität der Synchronisierung. Durch Erkenntnisse aus den aktuellen Daten der eingesetzten CA-Software ist zu erkennen, dass mehr qualifizierte und einfache Zertifikate als SSL-Zertifikate ausgestellt werden. Dadurch werden die entsprechenden Nodes mehr belastet als die restlichen.

Dieses Konzept bietet keine Ausfallsicherheit und keine Möglichkeit von Wartungsarbeiten und ist daher nur in Kombination mit einer der vorherigen Konzepte sinnvoll.

4.2.4 CA-Hierarchie mit Failover Cluster

Da eine CA-Hierarchie keine erhöhte Verfügbarkeit liefert, jedoch zu einer Lastverteilung führt, ist es sinnvoll, sie mit einem Failover Cluster zu kombinieren. Ein mögliches Umsetzungskonzept dieser Lösung ist in Abbildung 4.11 dargestellt. Dieses Konzept vereint die Vorteile des Failover Clusters mit jenen der CA-Hierarchien, bringt jedoch einen starken Anstieg der Kosten und einen erhöhten Wartungsaufwand mit sich.

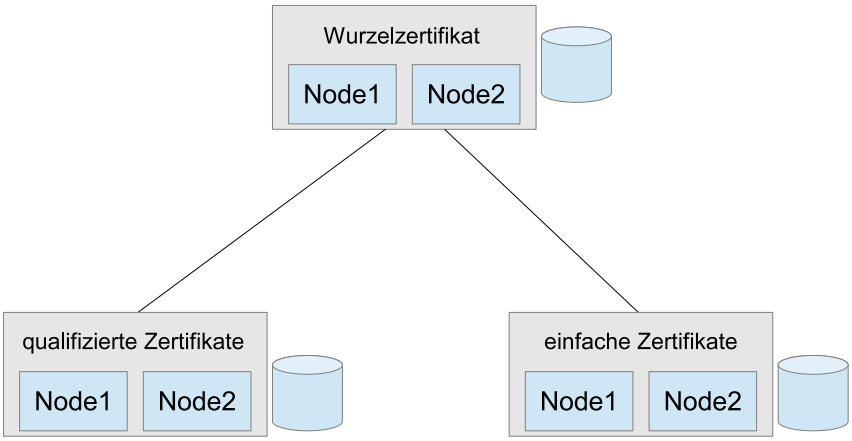


Abbildung 4.11: CA-Hierarchie mit Failover Cluster

4.2.5 Zusammenfassung

Eine Übersicht über die möglichen Techniken für Ausfallszenarien und Lastverteilung ist in Tabelle 4.2 dargestellt.

Name	erhöhte Verfügbarkeit	gleichmäßige Lastverteilung	Wartung im aktiven Betrieb	Softwareanpassungen[1]	Kosten[2]	Wartungsaufwand[3]	Erkennung fehlerhafter Services
Failover Cluster	✓	−−	✓				++
Loadbalancing	✓	++	✓	+	−	−	◯
CA-Hierarchie/Verteilung		◯		◯	−	+	−−
CA-Hierarchie mit Failover Cluster	✓	◯	✓	◯	++	++	++

++ sehr gut + gut ◯ zufriedenstellend − schlecht −− sehr schlecht

[1] Softwareanpassungen im Vergleich zu Failover Cluster (aktuelle Technologie)
[2] Kosten im Vergleich zu Failover Cluster (aktuelle Technologie)
[3] Wartungsaufwand im Vergleich zu Failover Cluster (aktuelle Technologie)

Tabelle 4.2: Zusammenfassung der Ausfallszenarien bzw. der Lastverteilung

Der Failover Cluster bzw. in Kombination mit einer CA-Hierarchie bietet die höchste Ausfallsicherheit. Dem Failover Cluster alleine fehlt jedoch die Möglichkeiten der Lastverteilung. Gegen die Kombination Failover Cluster mit einer CA-Hierarchie sprechen die stark erhöhten Kosten und der Wartungsaufwand.

Ein sinnvoller Mittelweg ist die Verwendung eines Loadbalancings, das eine modera-

te Fehlererkennung und eine gleichmäßige Lastverteilung bei geringeren Kosten und geringerem Wartungsaufwand bietet. Jedoch müssen in der Entwicklungsphase einige Punkte beachtet werden.

4.3 Zertifikatsformate und -regeln

Als Basis für dieses Kapitel dient die bereits durchgeführte Analyse in [Hag13, Kapitel 4.3]. Die dort bearbeiteten Varianten werden nachfolgend kurz beschrieben, zusätzlich werden weitere Möglichkeiten auf Grundlage der Analyse aus Kapitel 2.3 hinzugefügt.

4.3.1 SmartTrust-CA-Zertifikatsregeln

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.3.1] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Bei der SmartTrust-CA wird jeder Zertifikatsprozedur ein Zertifikatsformat zugeordnet. Dieses Zertifikatsformat ist eine Datei im Dateisystem, in der einfache Regeln beschrieben sind. Die Zertifikatsprozedur ist ein Datenbankeintrag, der die Verknüpfung zwischen Zertifikatsformat und CA-Zertifikat herstellt und einige Fixwerte für das zu erstellende Zertifikat enthält.

Wie in Listing 4.1 zu sehen ist, gibt es unterschiedliche Befehle, die über einen Index sortiert werden. Zusammengefasst ergeben sich folgende Operatoren:

- move (verschieben)
- copy (kopieren)
- set (setzen)
- remove (entfernen)
- rule (Operationen, z.B. + für Zeichenkettenverknüpfung)
- replace (ersetzen)
- encoding (ASN.1 Encoding festlegen)
- critical (Zertifikatserweiterung kritisch markieren)

Die Zertifikatsformate der SmartTrust-CA bieten sowohl die Möglichkeit, Fixwerte (Zertifikatsprocedure) festzulegen als auch Felder durch Regeln zu verändern.

```
[FormatDefinitionFields]
# encoding for certificate fields
CommonName.Encoding=UTF8String

FieldComposer.CommonName.rule = UniqueIdentifier.value +
   ":Firma"
FieldComposer.CommonName.replace=true
Extension.keyUsage.critical=true
FieldOperator.1.remove=LocalityName.value

# add rfc822name to subject alternative name extension
Fieldoperator.1.move=CommonName.value,GivenName.value
Extension.subjectaltname.attributes.0.othername.encoding=
   UTF8String
Extension.subjectaltname.attributes.0.othername.oid
   =1.3.6.1.4.1.311.20.2.3
```

Listing 4.1: Zertifikatsformat und Regeln der SmartTrust-CA

4.3.2 EJBCA-Zertifikatsregeln

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.3.2] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Die Zertifikatsregeln bei der EJBCA werden über ein Webinterface konfiguriert und in einer Datenbank gespeichert. Zusätzliche OIDs werden in einer Java Properties-Datei hinzugefügt. Aufwendigere Zertifikatserweiterungen können durch die Implementierung einer Java-Klasse (vgl. [EJB14a]) umgesetzt werden.

Das Hinzufügen zusätzlicher OIDs und der Java-Klassen kann nur durch technisches Personal vorgenommen werden. In beiden Fällen wird eine Anpassung an die Software durchgeführt. Ein Beispiel für das Zertifikatsformat der EJBCA ist in Anhang B enthalten.

Die im Webinterface gesetzten Werte entsprechen Fixwerten für den jeweiligen Zertifikatstyp (Zertifikatsformat). Nur durch die Erweiterung mit Java-Klassen können Operationen vergleichbar mit Zertifikatsregeln abgebildet werden.

4.3.3 OpenSSL-Zertifikatsformat

Wie in der Analyse (s. Kapitel 2.3) zu ersehen ist, verwenden die meisten CA-Systeme OpenSSL-Konfigurationsdateien für das Zertifikatsformat. Diese haben einen Aufbau ähnlich dem der Smarttrust-CA-Zertifikatsformate (s. Listing 4.2).

```
1  openssl_conf = openssl_init
2
3  [openssl_init]
4
5  [allgemein]
6  basicConstraints = CA:FALSE
7  keyUsage=critical, digitalSignature, keyEncipherment
8  subjectKeyIdentifier=hash
9  authorityKeyIdentifier=keyid,issuer
10 authorityInfoAccess=@aia_sect
11 certificatePolicies=ia5org,@polsect
12 crlDistributionPoints=@crl_section
13 subjectAltName=@altname
14 1.2.40.0.10.1.1.2=ASN1:NULL
15
16 [polsect]
17 policyIdentifier=1.2.40.0.17.1.11
18 CPS.1="http://www.a-trust.at/docs/cp/a-sign-test"
19
20 [aia_sect]
21 caIssuers;URI.1=http://www.a-trust.at/a-sign-test-03.crt
22 OCSP;URI.1=http://ocsp.a-trust.at/OCSP
23
24 [crl_section]
25 URI.1=ldap://ldap.a-trust.at/out=a-sign-SSL-03,o=A-Trust,
      c=AT?certificaterevocationlist?base?objectclass=
      eidCertificateAuthority
26
27 [altname]
28 otherName=1.3.6.1.4.1.311.20.2.3;UTF8:test@test.com
29 email=test@test.com
```

Listing 4.2: OpenSSL Zertifikatsformat

Der grundlegende Aufbau von OpenSSL-Erweiterungen besteht aus dem Extension-Namen, einem Kennzeichen für das Critical Flag und den Extension-Optionen, wie in Listing 4.3 dargestellt.

```
extension_name=[critical,] extension_options
```

Listing 4.3: Aufbau OpenSSL Extensions (vgl. [Ope14a])

Die Extensions in OpenSSL werden in zwei Kategorien eingeteilt, „Standard Extensions“ und „Arbitrary Extensions“. Standard Extensions sind von OpenSSL vordefinierte Zertifikatserweiterungen, wie z.B. `keyUsage`, `extKeyUsage` oder `subjectAltName`. Eine vollständige Liste der Standarderweiterungen ist in [Ope14a] aufgelistet. Arbitrary Extensions wird jede Zertifikatserweiterungen genannt, für die OpenSSL keinen Namen vordefiniert hat. Diese werden direkt über die OID der Erweiterung angegeben. Ein Beispiel dafür ist in Listing 4.4 zu sehen.

```
1.2.3.4=critical,ASN1:UTF8String:Some random data
```

Listing 4.4: Beispiel Arbitrary Extension (vgl. [Ope14a])

Mithilfe der OpenSSL-Konfigurationsdateien können auch beliebige Strukturen für den Wert der Zertifikatserweiterung erstellt werden. Dies wird wie in Listing 4.5 über Sections erreicht.

```
1.2.3.4=ASN1:SEQUENCE:seq_sect
[seq_sect]
field1 = UTF8:field1
field2 = UTF8:field2
```

Listing 4.5: Beispiel beliebiger Struktur in Zertifikatserweiterung (vgl. [Ope14a])

Wie zu erkennen ist, bietet OpenSSL nur die Möglichkeit, Fixwerte in den Konfigurationsdateien anzugeben. Zertifikatsregeln, welche Zertifikatsinhalte manipulieren, sind nicht möglich.

Synchronisierung, Signatur und Nachverfolgbarkeit

Um die Anforderung der Synchronisierung zwischen unterschiedlichen Standorten zu lösen, können die OpenSSL-Konfigurationsdateien auch in einer Datenbank abgelegt werden. Dadurch wäre eine Synchronisierung zwischen unterschiedlichen Standorten einfacher.

Die Anforderungen hinsichtlich Signatur und Nachverfolgbarkeit sind ebenfalls durch die Datenbank zu lösen. Eine Signatur der Datei wird zusätzlich in der Datenbank und alte Einträge werden in einem AuditLog gespeichert.

4.3.4 Aufbau von Zertifikaten via XML

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.3.3] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Durch die Ähnlichkeiten der ASN.1 Strukturen von Zertifikaten und der mit XML abbildbaren Strukturen besteht die Möglichkeit vollständige Zertifikat oder nur Zertifikatserweiterungen mittels XML abzubilden.

Die Grundstrukturen eines Zertifikats sind im RFC 5280 (s. auch Kapitel 2.1.3) festgelegt und unveränderbar, daher ist diese Technologie sinnvoll für die Zertifikatserweiterungen einsetzbar. Als Beispiel für eine oftmals benötigte Zertifikatserweiterung wird in Listing 4.6 der UPN für Windows Logon dargestellt:

```
1 <sequence>
2   <oid>1.3.6.1.4.1.311.20.2.3</oid>
3   <tagged tag='0'>
4     <utf8string>test@test.com</utf8string>
5   </tagged>
6 </sequence>
```

Listing 4.6: Zertifikatserweiterung UPN via XML

Ohne weitere Anpassungen unterstützt dieses Format nur Fixwerte. Es besteht keine Möglichkeit, Feldinhalte zu manipulieren.

4.3.5 Aufbau von Zertifikaten via YAML

Bei der r509 Ruby Certificate Authority (s. Kapitel 2.3.9) werden YAML Konfigurationsdateien für die Zertifikatsformate verwendet. Ein Beispiel dieser Dateien ist in Anhang E abgebildet.

Auf Basis der Konfigurationsdateien der r509 Ruby Certificate Authority und des Kapitels 4.3.4 kann ein Zertifikat wie in in Listing 4.7 abgebildet werden. Die Standard Zertifikatsfelder wie `version`, `serialNumber`, `algorithm` und `subject` sind in Zeile 2 bis 8 dargestellt. Diese werden als YAML-Assoziatives-Array eingefügt. Die Zertifikatserweiterungen (Zeile 8 - 20) werden als Unterpunkt des Eintrages `extensions` ebenfalls als Assoziatives-Array aufgelistet.

```
x509Certificate:
  version: 2
  serialNumber: 968908
  algorithm: "1.2.840.113549.1.1.5"
  subject:
    C: "AT"
    CN: "www.a-trust.at"
    OU: "Technik"
  extensions:
    keyUsage:
      critical: true
      value:
        - digitalSignature
        - keyEncipherment
    certificatePolicies:
      ciritical: false
      value:
        - policyIdentifer: 1.2.40.0.17.1.20
            cpsUri: http://www.a-trust.at/docs/cp/a-sign-
               ssl
            userNotice: ""
```

Listing 4.7: Einfacher Aufbau von Zertifikaten via YAML

Ebenso wie im XML Format ist es möglich, eine allgemeinere Form der YAML Datei zu erstellen, welche die ASN.1 Struktur des Zertifikates abbildet bzw. nur eine einzelne Extension. Die YAML Struktur hat jedoch gegenüber der XML Struktur den Nachteil, dass keine Unterscheidung zwischen Attributen und Inhalt möglich ist. Dies zeigt sich im Speziellen in Listing 4.8 in Zeile 4 bis 6. Das ASN.1 Object `tagged` benötigt einen `tag` Wert und kann beliebig viele weitere Objekte enthalten. Um dies in YAML abzubilden wird diesem Objekt ein `tag` und ein `value` Wert zugewiesen. Dies führt aber zu einer unterschiedlichen Kodierung als bei den restlichen ASN.1 Objekten.

```
smartcardLogonExtension:
  sequence:
    oid: "1.3.6.1.4.1.311.20.2.3"
    tagged:
      tag: 0
      value:
          utf8String: "test@test.com"
```

Listing 4.8: Zertifikatserweiterung UPN via YAML

4.3.6 Zertifikatsregeln in der Datenbank

Dieses Kapitel wurde bereits in [Hag13, Kapitel 4.3.4] detailliert besprochen, daher folgt hier nur eine Zusammenfassung.

Bei dieser Variante werden Operationen, welche ähnlich jener der SmartTrust-CA (s. Kapitel 4.3.1) sind, in Form von Datenbankeinträgen abgebildet. Dies ist beispielhaft in Tabelle 4.3 dargestellt.

Sort.	Operation	Feld	Wert
1	set	keyusage.critical	true
2	set	Subject.OrganisationalUnit	”Allg. beeidete/r und gerichtlich zertifizierte/r Dolmetscher/in”
3	asn1format	Subject.LocalityName	PrintableString
4	remove	Subject.CommonName	

Tabelle 4.3: Beispiel Zertifikatsregeln in der Datenbank

Durch eine Verknüpfung der Variante des Aufbaus von Zertifikaten via XML aus Kapitel 4.3.4 mit dieser Variante ergibt sich die Möglichkeit, Vorlagen für Zertifikatserweiterungen einzuführen. Dies ist in Tabelle 4.4 dargestellt.

Sort.	Operation	Feld	Wert
1	template	subjectaltname.1.othername	<sequence> <oid>1.2.5.39.37</oid> <utf8string> test@test.com </utf8string> </sequence>

Tabelle 4.4: Beispiel Zertifikatsregeln in der Datenbank mit XML Vorlagen

Die Speicherung der Zertifikatsformate und -regeln in einer Datenbank anstelle des Dateisystems bringt den Vorteil der einfachen Sychronisierung zwischen einzelnen Servern und Standorten. Durch eine Unterschrift von zwei Security Officer und Ablage dieser Signatur in der Datenbank wird der Anforderung der Nachverfolgbarkeit entsprochen.

4.3.7 Zertifikatsregeln mittels Scriptsprachen

Für die Manipulation von statischen Grundstrukturen werden oft Scriptsprachen eingesetzt. Eines der bekanntesten und weitverbreitetsten Beispiele ist der Einsatz von

Javascript um dynamische Inhalte in HTML-Webseiten zu erreichen. Dieses Konzept dient als Grundlage für die Zertifikatsregeln mittels Scriptsprache.

Durch den Einsatz von z.B. Javascript oder Lua[9] können statische Zertifikatsformate abgeändert werde. Als Beispiel einer solchen Kombination aus statischen Zertifikatsformat und Scriptsprache wird nachfolgend die Kombination XML (s. Kapitel 4.3.4) und Javascript verwendet. Jedoch sind auch Kombinationen von YAML, JSON oder OpenSSL-Konfigurationsdateien mit Javascript, Lua oder Python möglich.

Wie bereits in [Hag13, Kapitel 4.3.3] dargestellt, kann die gesamte Struktur eines Zertifikates in XML ausgedrückt werden. Dies ist in Listing 4.9 dargestellt.

```
<?xml version='1.0'?>
<!DOCTYPE sequence[
  <!ATTLIST sequence id ID #IMPLIED>
  <!ATTLIST tagged id ID #IMPLIED>  ]>
<sequence>
  <sequence>
    <tagged tag='0'> <!-- version -->
      <integer>2</integer>
    </tagged>
    <integer>968907</integer> <!-- serialNumber -->
    <sequence> <!-- signingAlgorithm -->
      <oid>1.2.840.113549.1.1.5</oid>
      <null>
    </sequence>
    <sequence id='subject'>
      <set>
        <sequence>
          <oid>2.5.4.3</oid>
          <UTF8String>www.test.com<UTF8String>
        </sequence>
      </set>
      ...
    </sequence>
    ...
    <tagged id='certExtensions' tag='3'>
      <sequence id='keyUsage'>
        <oid>2.5.29.15</oid>
        <OctetString>
          <BitString>11</BitString>
        </OctetString>
      </sequence>
```

[9] http://www.lua.org/

```
32          ...
33        </tagged>
34      </sequence>
35      ...
36  </sequence>
```

Listing 4.9: Grundstruktur XML-Zertifikatsformat

Für die bessere Verarbeitung dieser Grundstruktur wurden XML-id-Attribute eingefügt, sodass direkt auf die benötigten Elemente zugegriffen werden kann. Damit diese id-Attribute auch vom Javascript XML Parser interpretiert werden, müssen zusätzlich die DTD[10]-Regeln in Zeile 2-4 eingefügt werden.

Die Scriptsprache hat die Aufgabe der Zertifikatsregeln mit diesen werden die Inhalte der Grundstruktur dynamisch angepasst. In Listing 4.10 ist die Anpassung der Zertifikatserweiterung `keyUsage` durch ein Javascript-Programm dargestellt.

```
1  var parser = new DOMParser();
2  var doc = parser.parseFromString(strXML,"text/xml");
3
4  var keyUsage = doc.getElementById("keyUsage");
5  var bitString =
6        keyUsage.getElementsByTagName("BitString")[0];
7  bitString.innerHTML = "101";
8
9  var serializer = new XMLSerializer();
10 var strXml2 = serializer.serializeToString(doc);
```

Listing 4.10: Anpassung der XML Grundstruktur mit Javascript

In den ersten beiden Zeilen wird das XML Grundgerüst (Variable `strXML`) in ein XML-DOM[11]-Objekt umgewandelt. Dieses wird zur weiteren Manipulation der XML-Daten verwendet. In Zeile 4 wird zuerst das Element mit der Id `keyUsage` gewählt, anschließend wird ein untergeordnetes Element mit dem Namen `BitString` ausgesucht (Zeile 5-6) und der Inhalt geändert (Zeile 7). Zeilen 9-10 beinhalten die Umwandlung des DOM-Objektes zurück in eine Zeichenkette für die weitere Verarbeitung.

Durch den Einsatz von Javascript ist in den Zertifikatsregeln eine komplexere Logik, im Vergleich zu den bisher vorgestellten Varianten, möglich. Jedoch besteht damit auch die Gefahr, eine nicht in ASN.1 umsetzbare XML-Struktur zu erzeugen.

Für die Bearbeitung der Zertifikatsregeln sind grundlegende Programmierfähigkeiten

[10] Document Type Definition
[11] Document Object Model

und eine genaue Kenntnis der ASN.1 Struktur von Zertifikaten erforderlich. Beides kann nicht bei jedem Security Officer vorausgesetzt werden.

Ähnlich der Lösung für Synchronisierung, Signatur und Nachverfolgbarkeit wie in Kapitel 4.3.3 beschrieben, können auch die Scripte signiert und in der Datenbank gespeichert werden.

4.3.8 Zusammenfassung

Eine Übersicht über die möglichen Techniken für Zertifikatsformate und -regeln ist in Tabelle 4.5 dargestellt. Die OpenSSL-Zertifikatsregeln sind hier doppelt vorhanden, einmal mit Speicherung im Dateiformat, das zweite Mal mit einer Speicherung in der Datenbank. Die Varianten „Aufbau von Zertifikaten via XML“ bzw. „Aufbau von Zertifikaten via YAML“ wurden nicht vollständig beschrieben, weshalb bei diesen in Tabelle 4.5 in einigen Spalten der Wert „nicht anwendbar“ (n.a.) angeführt wird.

In den Anforderungen in Kapitel 3.2 Absatz „Zertifikatsformate und -regeln“ werden für die Zertifikatsformate Möglichkeiten für die Nachvollziehbarkeit und Protokollierung gefordert. Beide Bedingungen können durch organisatorische Maßnahmen in allen diskutierten Varianten umgesetzt werden. Um eine kryptographisch gesicherte Nachvollziehbarkeit zu erreichen, ist es jedoch sinnvoll, eine digitale Signatur über die Zertifikatsformate zu erstellen und diese parallel zu den Formaten zu speichern. Durch die Signatur kann zu einem späteren Zeitpunkt festgestellt werden, wer das Format geändert hat und zu welchem Zeitpunkt. Theoretisch ist die Einführung einer Signatur für alle diskutierten Zertifikatsformate möglich, per Design ist diese in den OpenSSL-, den SmartTrust- und den EJBCA-Formaten jedoch nicht vorhanden. Im Gegensatz dazu unterstützen XML-Dateien ein eigenes Signaturformat[12], das zusätzlich die Lesbarkeit der XML-Datei nicht beeinflusst[13].

In den Anforderungen ist die automatische Synchronisierung der Zertifikatsformate und -regeln über mehrere Server als eines der zu lösenden Probleme angeführt. Eine automatische Synchronisierung der Zertifikatsformate ist am einfachsten über eine Datenbankreplikation oder eine gemeinsame Datenbank zu erreichen. Dabei dürfen jedoch keine Formate im Dateisystem des jeweiligen Servers abgelegt werden. Die diskutierten Formate von OpenSSL, SmartTrust und EJBCA speichern die Formate als Datei und legen in der Datenbank nur eine Verknüpfung mit Statusinformationen an. Zertifikatsformate, die im Dateisystem abgespeichert werden, müssen über einen zusätzlichen Prozess auf alle Server verteilt werden. Dadurch ergeben sich weitere Probleme, wie z.B. Atomarität und Konsistenz. Ein weiterer Ansatz zur Lösung der Problematik der dateibasierten Zertifikatsformate zeigt die Variante „OpenSSL-Zertifikatsregeln + Datenbank“.

[12] XMLDSig, http://www.w3.org/TR/xmldsig-core/

[13] Verwendung von Enveloped Signature Transform
http://www.w3.org/TR/xmldsig-core/#sec-EnvelopedSignature

Name	signierte Zertifikatsformate	rückverfolgbare Zertifikatsformate	Speicherung in Datenbank	Speicherung in Dateisystem	automatische Synchronisation möglich	Möglichkeiten f. unbekannte Zertifikatserweiterungen	unterstützt alle Zertifikats-Subject-Felder	dynamische Inhalte durch Regeln
SmartTrust-CA-Zertifikatsregeln			✓	✓		◯	✓	✓
EJBCA-Zertifikatsregeln			✓	✓		◯		✓
OpenSSL-Zertifikatsformat				✓		+		
OpenSSL-Zertifikatsformat + Datenbank		✓	✓		✓	+		
Aufbau von Zertifikaten via XML	n.a.	n.a.	n.a.	n.a.	n.a.	++	✓	
Aufbau von Zertifikaten via YAML	n.a.	n.a.	n.a.	n.a.	n.a.	+	✓	
Zertifikatsregeln in der Datenbank	✓	✓	✓		✓	++	✓	✓
Zertifikatsregeln mittels Scriptsprachen	✓	✓	✓		✓	++	✓	✓

n.a. . . . nicht angegeben/anwendbar
++ sehr gut + gut ◯ zufriedenstellend − schlecht −− sehr schlecht

Tabelle 4.5: Zusammenfassung der Zertifikatsformate und -regeln

Hierbei werden die Vorteile der OpenSSL-Zertifikatsformate mit der automatischen Synchronisierung kombiniert.

4.4 Sperrlisten (CRL)

4.4.1 Berechnung der CRL-Zeiten

Aufgrund der Anforderungen zur Berechnung der CRL-Zeiten aus Kapitel 3.2 Paragraph „Sperrlisteninformationen (CRL)“ ergibt sich für einen Normalfall eine CRL-Erstellung, wie sie in Abbildung 4.12 dargestellt ist (s. auch [Ofe06, S. 26]).

Wie Abbildung 4.12 zu erkennen gibt, wird das Gültig-bis-Datum der CRL durch die

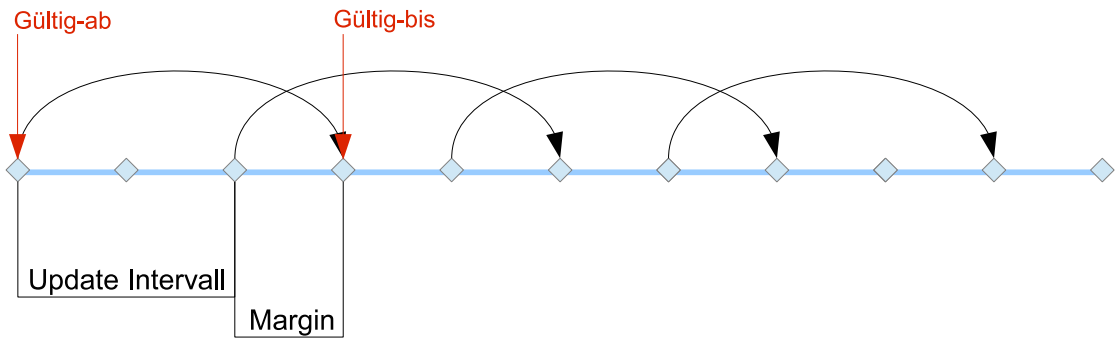


Abbildung 4.12: CRL-Zeiten – Normalfall

Formel 4.1 errechnet.

$$\text{Gültig-bis} = \text{Gültig-ab} + \text{Update Interval} + \text{Margin} \tag{4.1}$$

Der Zeitpunkt der Erstellung der nächsten CRL kann über zwei Formeln berechnet werden. Einmal erfolgt dies auf der Basis des Gültig-ab-Datums, indem das Update-Intervall addiert wird, wie in Formel 4.2 dargestellt. Die zweite Möglichkeit besteht darin, vom errechneten Gültig-bis-Datum die Margin-Zeit zu subtrahieren; dies ist in Formel 4.3 dargestellt.

$$\text{neues Gültig-ab} = \text{altes Gültig-ab} + \text{Update Interval} \tag{4.2}$$

$$\text{neues Gültig-ab} = \text{altes Gültig-bis} - \text{Margin} \tag{4.3}$$

Die in der nachfolgenden Liste enthaltenen Fälle führen zu einem veränderten Ausstellungszeitpunkt der Sperrliste.

- Sofortausstellung (immediate issue)
- Intervall verlängern/verkürzen
- CA schließen/öffnen mit jeweils keinen/einer/mehreren verlorenen CRLs

Bei diesen Fällen ist zu beachten, wie sich das Ausstellungsdatum (Gültig-ab-Datum) der Sperrliste auf das Gültig-bis-Datum auswirkt.

4.4.1.1 Verschiebung des Gültig-bis-Datums

Wird bei der Erstellung einer neuen Sperrliste außerhalb der normalen Intervalle (z.B. Sofortausstellung) so wie in Formel 4.1 vorgegangen, als werden sowohl das Update-Intervall als auch die Margin-Zeit zu dem Gültig-ab-Datum hinzugezählt, führt dies zu einer Verschiebung der Gültig-bis-Daten wie in Abbildung 4.13 dargestellt.

Durch dieses Verhalten ist der Zeitpunkt der Erstellung der nächsten Sperrliste nicht vorausberechenbar. Vor allem für Client-Programme, welche die Sperrlisten in regelmä-

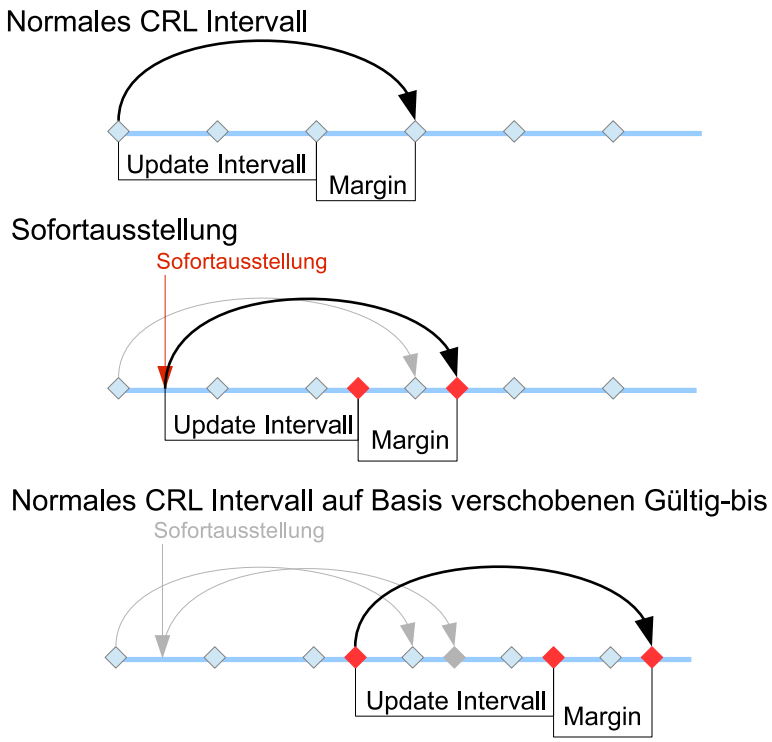


Abbildung 4.13: CRL-Zeiten – Verschiebung des Gültig-bis-Datums

ßigen Abständen prüfen, kann es zu deutlich verkürzten Gültigkeitszeiten der Sperrliste kommen. Dies ist z.B. der Fall, wenn eine Sofortausstellung um die Margin-Zeit plus wenige Minuten nach der letzten CRL-Erstellung stattfindet.

$$
\begin{aligned}
&\text{Sofortausstellung} = \text{altes Gültig-ab} + \text{Margin} + \varepsilon \\
&\text{neues Gültig-ab} = \text{Sofortausstellung} + \text{Update Intervall} \\
&\\
&\text{z.B.: altes Gültig-ab} = 12{:}00{:}00 \\
&\text{Update-Intervall} = 3h;\ \text{Margin} = 1h;\ \varepsilon = 5min \\
&\\
&\text{altes Gültig-bis} = 12{:}00{:}00 + 3h + 1h = 16{:}00{:}00 \\
&\text{Sofortausstellung} = 12{:}00{:}00 + 1h + 5min = 13{:}05{:}00 \\
&\text{neues Gültig-ab} = \text{Sofortausstellung} = 13{:}05{:}00 \\
&\text{neues Gültig-bis} = 13{:}05{:}00 + 3h + 1h = 17{:}05{:}00 \\
&\\
&\text{CRL 1} = 12{:}00{:}00 \text{ bis } 16{:}00{:}00 \\
&\text{CRL 2} = 13{:}05{:}00 \text{ bis } 17{:}05{:}00 \\
&\text{CRL 3} = 16{:}05{:}00 \text{ bis } 20{:}05{:}00
\end{aligned}
\tag{4.4}
$$

Wie in der Berechnung 4.4 zu sehen ist, liegen die Gültig-bis-Daten der CRL 1 und CRL 2 nur 1 Stunde und 5 Minuten auseinander.

Ein Client-Programm würde zuerst eine 3-Stunden-CRL[14] herunterladen, nach deren Ablauf eine 1-Stunde-und-5-Minuten-CRL. Die nächste CRL hätte wieder die volle Länge von 3 Stunden. Dies führt zu kürzeren Sperrlisten-Gültigkeitszeiten am Client und somit zu häufigerem Nachladen der aktuellen CRL.

Ein weiterer Nachteil ist die Beeinflussbarkeit der Sperrlistenzeiten durch Kunden. Durch gezielte Widerrufe und der damit verbundenen Sofortausstellungen kann ein potentieller Angreifer die Erstellungszeiten der Sperrlisten so beeinflussen, dass diese annähernd gleichzeitig stattfinden, um somit eine erhöhte Last zu einem gewünschten Zeitpunkt herbeiführen.

4.4.1.2 Berechenbares Gültig-bis-Datum

Um das Gültig-bis-Datum für Client-Programme vorausberechenbar zu machen, wird es bei der Erstellung einer außerplanmäßigen Sperrliste beibehalten. Dies ist in Abbildung 4.14 dargestellt.

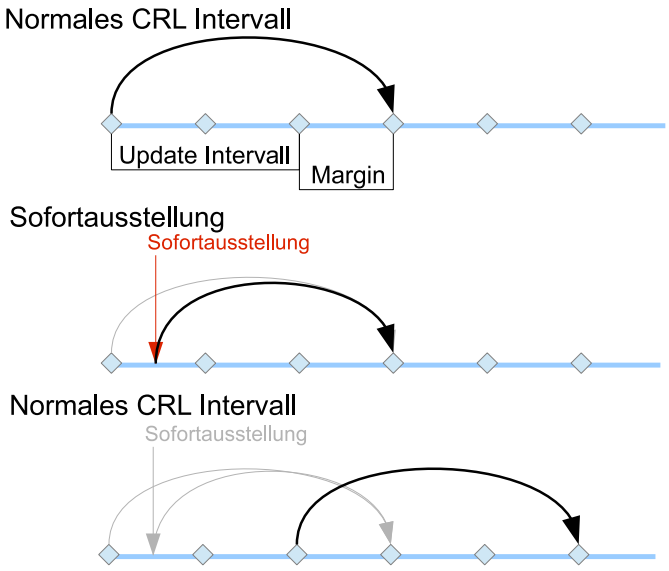


Abbildung 4.14: CRL-Zeiten – Berechenbares Gültig-bis-Datum

Dadurch werden Sperrlisten ausgestellt, die insgesamt eine kürzere Gültigkeit haben, jedoch mit dem gleichen Gültig-bis-Datum wie die vorherige reguläre Sperrliste. Um dieses Verhalten zu erzielen, muss das Gültig-ab-Datum der nächsten CRL über das Gültig-bis-Datum der aktuellen CRL berechnet werden; die zugehörige Formel ist in 4.3 dargestellt.

[14] Auf der Basis einer zuvor erhaltenen CRL und der Margin-Zeit

4.4.1.3 Nicht reguläre Sperrlisten

Auf der Basis der Erkenntnisse aus Kapitel 4.4.1.2 ergeben sich in weiterer Folge die CRL-Daten, wie in den folgenden Absätzen beschrieben wird.

Sofortausstellung

Die in den Anforderungen beschriebene Sofortausstellung (immediate issue) von CRLs führt zu einer verkürzten Gültigkeit der Sperrliste. Ein Ablauf ist in Abbildung 4.15 dargestellt. Wie in Kapitel 4.4.1.2 beschrieben, bleibt der Wert für das Gültig-bis-Datum der Sperrliste bestehen, nur das Gültig-ab-Datum wird angepasst.

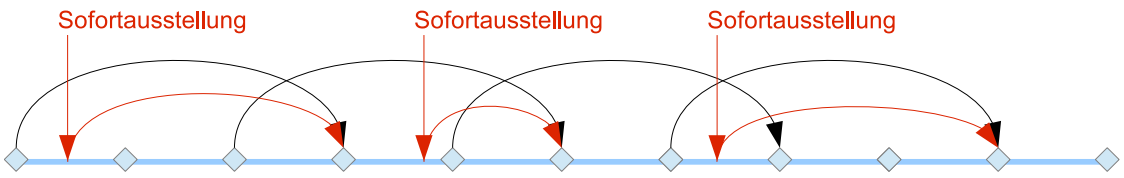


Abbildung 4.15: CRL-Zeiten – Sofortausstellung

Verlängerung/Verkürzung des Intervalls

Bei einer Anpassung des Update-Intervalls der Sperrlisten wirkt sich dieses erst auf die nächste Sperrliste aus. Aufgrund der Intervall-Änderung wird keine neue Sperrliste erstellt. In den folgenden Abbildungen 4.16 und 4.17 sind jeweils eine Verlängerung des Update-Intervalls und eine Verkürzung dargestellt.

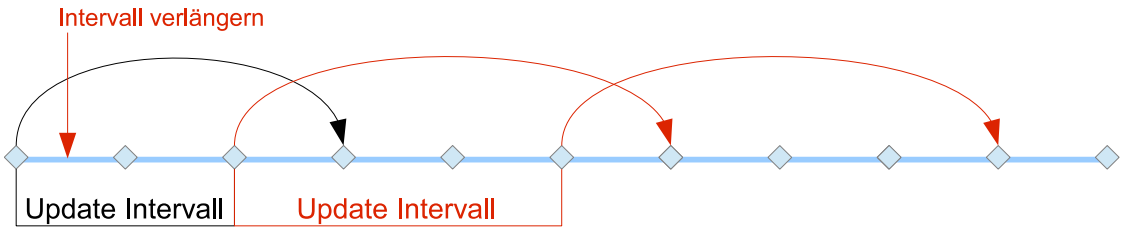


Abbildung 4.16: CRL-Zeiten – Intervall verlängern

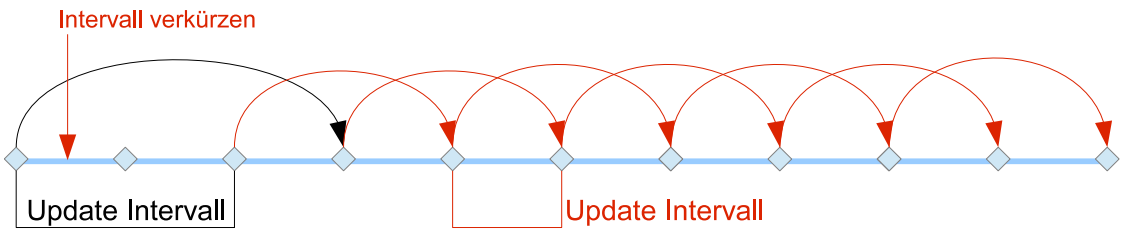


Abbildung 4.17: CRL-Zeiten – Intervall verkürzen

Verlängerung/Verkürzung der Margin-Zeit

Ebenso wie bei der Anpassung des Update-Intervalls wirkt sich auch eine Veränderung der Margin-Zeit nicht auf die aktuelle CRL aus. Erst bei der nächsten Sperrliste wird die Veränderung sichtbar. In den Abbildungen 4.18 und 4.19 sind jeweils eine Verlängerung und eine Verkürzung der Margin-Zeit dargestellt.

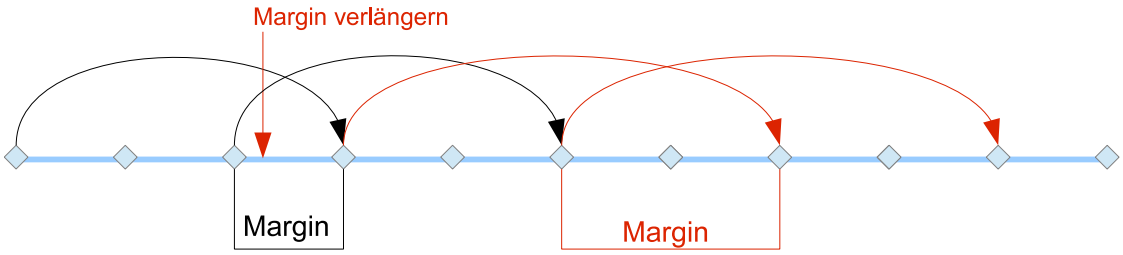


Abbildung 4.18: CRL-Zeiten – Margin-Zeit verlängern

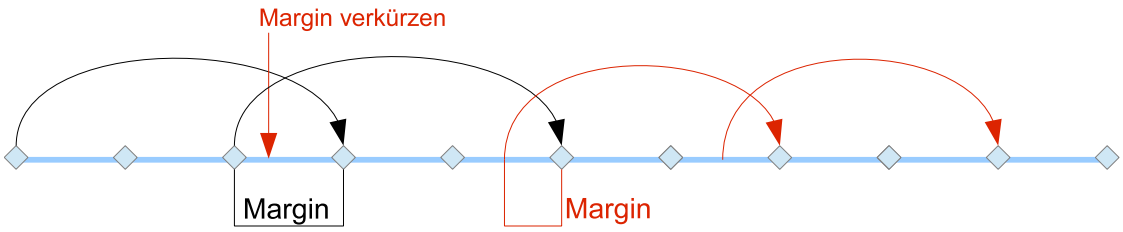


Abbildung 4.19: CRL-Zeiten – Margin-Zeit verkürzen

Schließen und Öffnen einer CA

Beim Schließen einer CA wird keine neue Sperrliste erstellt, beim anschließenden erneuten Öffnen wird sofort die erste Sperrliste erstellt. Dadurch kann es zu einigen Fällen kommen, bei denen keine (Abb. 4.20), eine (Abb. 4.21) oder mehrere (Abb. 4.22) Sperrlisten in der Zeit zwischen Schließen und Öffnen nicht erstellt wurden. Diese Fälle finden auch Anwendung, wenn die Sperrliste aus anderen Gründen nicht erstellt werden kann und erst nach einiger Zeit wieder aktualisiert wird.

Damit für die Betrachtungsweise des Client-Programms das gleiche CRL-Intervall bestehen bleibt, werden die Zeiten von nicht ausgestellten Sperrlisten berechnet, um dadurch das weitergerechnete Gültig-bis-Datum zu erfahren. Die Abbildungen 4.20, 4.21 und 4.22 zeigen die Berechnung der CRL-Zeiten für die eingangs beschriebenen Fälle.

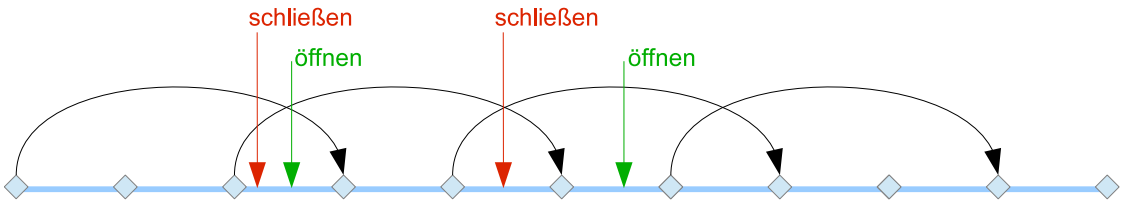


Abbildung 4.20: CRL-Zeiten – CA schließen/öffnen – keine CRL ausgefallen

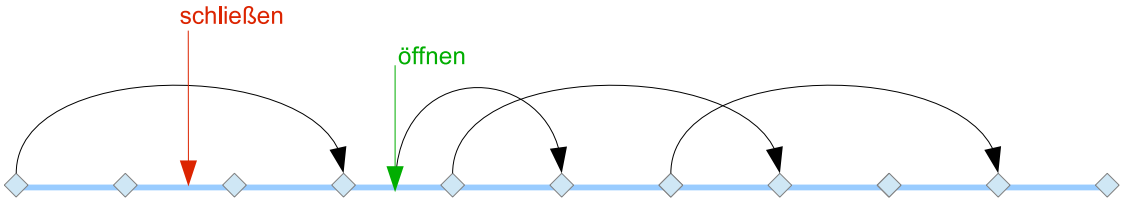


Abbildung 4.21: CRL-Zeiten – CA schließen/öffnen – eine CRL ausgefallen

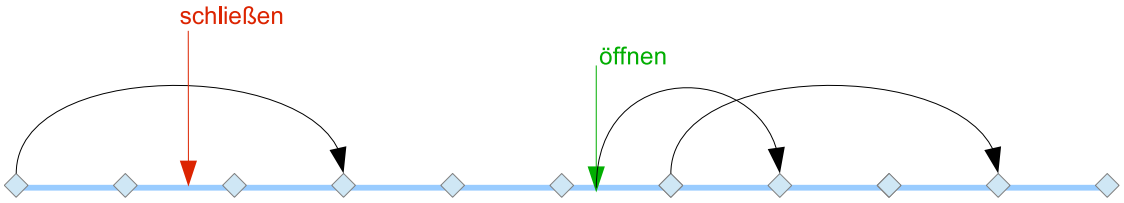


Abbildung 4.22: CRL-Zeiten – CA schließen/öffnen – mehrere CRLs ausgefallen

4.4.2 Größenprobleme bei Sperrlisten

Sperrlisten sind wie in Kapitel 2.1.4 beschrieben aufgebaut. Sie bestehen aus Standardfeldern, die sich zwischen den einzelnen Sperrlisten nur minimal ändern, und den Einträgen der widerrufenen Zertifikate, die den Großteil der Dateigröße ausmachen. In Anhang G wurde eine Formel entwickelt, wie die Größe einer Sperrliste mit einer bestimmten Anzahl an Einträgen annäherungsweise berechnet werden kann. Die potenziell maximale Anzahl der Zertifikatseinträge in einer Sperrliste sind die von der zugehörigen CA ausgestellten Zertifikate.

Bei den Größenproblemen sind zwei unterschiedliche Sichtweisen zu betrachten. Zum einen treten durch große Sperrlisten Probleme bei der Erstellung und Verteilung in der CA-Software auf. Das zweite Problem ist die Zeit, die eine Client-Software benötigt, um die großen Sperrlisten herunterzuladen und zu interpretieren bzw. zu durchsuchen. Um diese Größenprobleme zu lösen, sind mehrere Möglichkeiten vorstellbar.

Partitioned CRLs

Partitioned CRLs basieren auf dem im RFC 5260 beschriebenen Eintrag für die CRL Distribution Points (s. [DC08, Kapitel 4.2.1.13. CRL Distribution Points]). Darin wird zu einer CRL-URI ein zusätzlicher Wert „reason“ angegeben. Ist dieser Wert angegeben,

enthält die Sperrliste unter dieser URI nur die Widerrufe für diesen Sperrcode. Durch diesen Mechanismus wird eine große Sperrliste in mehrere kleine Sperrlisten aufgeteilt.

Die Größenunterschiede zwischen einer gesamten Sperrliste und den aufgeteilten Sperrlisten wird anhand zweier Sperrlisten in Tabelle 4.6 dargestellt. Zur Berechnung der Größen der aufgeteilten Sperrlisten wird die Formel aus Anhang G verwendet.

Name	**Sperrliste** premium enc 02[1]	**Sperrliste** premium sig 02[2]
Anzahl Einträge (gesamt)	87.989	107.684
Anzahl Einträge (superseeded)	39.470	39.934
Anzahl Einträge (cessationOfOperation)	45.744	46.301
Anzahl Einträge (keyCompromise)	2.767	20.638
Anzahl Einträge (certificateHold)	8	811
Größe gesamt	3.168.136,00 bytes 3,02 MB	3.877.156,00 bytes 3,60 MB
Größe CRL superseeded	1.421.699,00 bytes 1,36 MB	1.438.403,00 bytes 1,37 MB
Größe CRL cessationOfOperation	1.647.563,00 bytes 1,57 MB	1.667.615,00 bytes 1,59 MB
Größe CRL keyCompromise	100.391,00 bytes 98,03 KB	743.747,00 bytes 726,32 KB
Größe CRL certificateHold	1.067,00 bytes 1,04 KB	29.975,00 bytes 29,27 KB

[1] a-sign-premium-enc-02
[2] a-sign-premium-sig-02

Tabelle 4.6: Sperrlistengröße gesamt und Partitoned CRL

Dieser Lösungsansatz beseitigt serverseitige Probleme. Der Erstellungsprozess und der Verteilungsprozess der Sperrlisten wird durch die kleineren Sperrlisten erleichtert. Auf der Client-Seite müssen jedoch mehrere Sperrlisten geladen werden. Für den Client ergeben sich dadurch keine nennenswerten Vorteile.

Delta-CRLs

Bei Delta-CRLs werden gesamte Sperrlisten nur in größeren Abständen erstellt (z.B. einmal täglich), während für Aktualisierungen zwischen diesen Gesamt-CRLs in kürzeren Abständen Delta-CRLs generiert werden. Diese enthalten nur jene Zertifikate, die sei der letzten Gesamt-CRL widerrufen wurden.

Tabelle 4.7 zeigt den Größenunterschied für den Abschnitt eines Tages einer normalen CRL (Gesamt-CRL oder Full-CRL) im Vergleich zu einer Delta-CRL. Es werden alle zwei Stunden reguläre Sperrlisten erstellt, bei der Delta-CRL wird einmal am Tag eine

Datum/Uhrzeit	**Full / Delta CRL**		**Full CRL**	
	Einträge	Größe	Einträge	Größe
15.05.2014 00 Uhr	104.074	2.706.703 bytes 2,58 MB	104.074	2.706.703 bytes 2,58 MB
15.05.2014 02 Uhr	4	883 bytes	104.078	2.706.807 bytes
15.05.2014 04 Uhr	5	909 bytes	104.079	2.706.833 bytes
15.05.2014 06 Uhr	21	1.325 bytes	104.095	2.707.249 bytes
15.05.2014 08 Uhr	30	1.559 bytes	104.104	2.707.483 bytes
15.05.2014 10 Uhr	141	4.445 bytes	104.215	2.710.369 bytes
15.05.2014 12 Uhr	146	4.575 bytes	104.220	2.710.499 bytes
Sperrliste gleichbleibend für 14 Uhr, 16 Uhr, 18 Uhr, 20 Uhr, 22 Uhr				
15.05.2012 00 Uhr	63.005	1.638.909 bytes 1,56 MB	63.005	1.638.909 bytes 1,56 MB
15.05.2012 02 Uhr	1	805 bytes	63.006	1.638.909 bytes
15.05.2012 04 Uhr	1	805 bytes	63.006	1.638.909 bytes
15.05.2012 06 Uhr	1	805 bytes	63.006	1.638.909 bytes
15.05.2012 08 Uhr	4	883 bytes	63.009	1.639.013 bytes
15.05.2012 10 Uhr	44	1.923 bytes	63.049	1.640.053 bytes
15.05.2012 12 Uhr	54	2.183 bytes	63.059	1.640.313 bytes
15.05.2012 14 Uhr	55	2.209 bytes	63.060	1.640.339 bytes
Sperrliste gleichbleibend für 16 Uhr, 18 Uhr, 20 Uhr, 22 Uhr				

Tabelle 4.7: Sperrlistengröße gesamt und Delta CRL (a-sign-premium-sig-02)

Full-CRL erstellt. Um die Übersichtlichkeit des Beispiels zu wahren, werden Sofortausstellungen der CRLs nicht berücksichtigt. Für den Vergleich werden zwei Tage herangezogen, einmal der 15. Mai 2014 und einmal der 15. Mai 2012. Im Fall der Delta-CRL werden von einem Client-Programm an diesem Tag 2,62 MB bzw. 1,58 MB Sperrlisten (Full-CRL + 11x Delta-CRLs) heruntergeladen. Im Fall der normalen CRL werden von einem Client-Programm an diesem Tag insgesamt 31,00 MB bzw. 18,77 MB (12x Full-CRL) heruntergeladen.

Bei dieser Variante liegt der Vorteil sowohl auf der Serverseite als auch auf der Client-Seite. Der Client muss trotz der Delta-CRL einmal die gesamte CRL herunterladen, kann jedoch anschließend mit den viel kleineren Delta-CRLs jeweils den aktuellen Status der Sperrliste erstellen.

Das Programm zum Interpretieren und Aufteilen der Sperrliste wurde im Zuge der Arbeit entwickelt und ist auf der beiliegenden CD-ROM im Ordner `/CRL_Größenprobleme/CrlToCsv/` enthalten, ebenso das zur Berechnung verwendete Microsoft-Excel-Dokument (s. Anhang H).

Neues CA-Zertifikat

Durch die Erstellung eines neuen CA-Zertifikats wird gleichzeitig ein neuer CRL Distribution Point eingeführt. Die aktuelle Sperrliste wird in ihrer Größe nicht verändert, alle neu ausgestellten Zertifikate verweisen jedoch auf eine neue kleinere CRL. Diese Variante bringt Vorteile für Server und Client, jedoch werden bestehende Zertifikate nicht berücksichtigt. Probleme ergeben sich durch die Veröffentlichung und Verteilung des neuen CA-Zertifikats.

Neuer CRL Distribution Point

Um die zusätzlichen Probleme, die durch ein neues CA-Zertifikat entstehen, zu verhindern, kann ein neuer CRL Distribution Point auch zu einem bestehenden CA-Zertifikat hinzugefügt werden. Alle neu ausgestellten Zertifikate verweisen auf den neuen CRL Distribution Point, das CA-Zertifikat muss in diesem Fall jedoch zwei Sperrlisten erstellen. Zusätzlich muss bei jedem ausgestellten Benutzerzertifikat mit abgespeichert werden, in welchen CRL Distribution Point es aufzunehmen ist, falls es widerrufen wird. Ebenso wie bei der vorhergehenden Variante ist auch hier der Vorteil auf Server- und Client-Seite.

OCSP-Abfragen forcieren

OCSP-Abfragen sind eine Möglichkeit, den Status eines einzelnen Zertifikats zu erfahren, ohne eine gesamte Sperrliste herunterzuladen. Durch diese Variante ergibt sich ein erheblicher Vorteil für die Client-Software, da eine OCSP-Abfrage um vieles kleiner als die Sperrliste ist. Für den Server ergeben sich dadurch keine Vorteile. Durch das Forcieren von OCSP-Abfragen sollen Entwickler von Client-Software davon überzeugt werden, für die Zertifikatsprüfung ihrer Software auf OCSP-Abfragen umzusteigen und Sperrlistenabfragen als Backup-Lösung einsetzen. Inwieweit dieses Vorgehen erfolgreich ist, hängt von den Entwicklern bzw. Kunden ab.

Keine CRL-Erstellung

Ein weiterer Ansatz, OSCP-Abfragen zu forcieren und Entwickler zum Umstieg zu zwingen, ist es, in zukünftigen Zertifikaten keinen CRL Distribution Point mehr aufzunehmen. Dadurch können Client-Programme die Zertifikatsprüfungen nur noch über OCSP-Abfragen durchführen. In weiterer Folge muss der Server für diese CA-Zertifikate keine Sperrlisten mehr erstellen und verteilen.

Zusammenfassung

Eine Übersicht über die möglichen Problemlösungsvarianten der Sperrlistengröße ist in Tabelle 4.8 dargestellt.

Name	rückwirkend möglich	Vorteile für Client-Software	clientseitige Unterstützung notwendig	Sperrliste verkleinert
Partitioned CRL		+	✓	◯
Delta-CRL		◯	✓	+
neues CA-Zertifikat		++		++
neuer CRL Distripution Point		++		++
OCSP-Abfrage forcieren	✓	++	✓	−−
keine CRL-Erstellung		++	✓	++

++ sehr gut + gut ◯ zufriedenstellend
− schlecht −− sehr schlecht

Tabelle 4.8: Zusammenfassung der Vorschläge zur Problemlösung der Sperrlistengröße

Wie in Kapitel 4.8 zu erkennen ist, wird durch die Varianten „Partitioned CRL“ und „Delta-CRL“ nur ein geringer Vorteil erreicht. Zudem wird eine clientseitige Unterstützung vorausgesetzt. Von den diskutierten Varianten sind ein „neues CA-Zertifikat“ und ein „neuer CRL Distribution Point“ diejenigen, welche die beste Lösung des Problems bieten, ohne dabei eine Änderung der Client-Software erforderlich zu machen. Jedoch wird das Problem dadurch nur für zukünftig ausgestellte Zertifikate gelöst. Vielversprechend ist eine Kombination einer der beiden Methoden mit der Variante „OCSP-Abfragen forcieren“, da dadurch auch ein Vorteil für die bereits ausgestellten Zertifikate gegeben wäre.

4.5 OCSP-Server

Der aktuelle OCSP-Server wurde nach dem mittlerweile ersetzten RFC 2560[15] entwickelt. In diesem sind die erlaubten Signatoren für die OCSP-Antwort wie folgt beschrieben (s. [MM99, Kapitel 4.2.2.2, Authorized Responders]):

[15] http://tools.ietf.org/html/rfc2560

„They [Systeme und Clients, die OCSP-Anfragen benutzen; Anmerk. d. Verf.] MUST reject the response if the certificate required to validate the signature on the response fails to meet at least one of the following criteria:

1. Matches a local configuration of OCSP signing authority for the certificate in question; or
2. Is the certificate of the CA that issued the certificate in question; or
3. Includes a value of id-ad-ocspSigning in an ExtendedKeyUsage extension and is issued by the CA that issued the certificate in question.“

Der aktuelle OCSP-Server verwendet die unter 1. beschriebene Variante eines Antwort-Signers. In weiterer Folge wird nur ein Zertifikat für OCSP-Antworten verwendet, das alle OCSP-Antworten signiert. In der aktuellen Version des RFC 6960[16] ist zusätzlich die folgende Notiz eingefügt worden (s. [MM13, Kapitel 4.2.2.2, Authorized Responders]):

„Note: For backwards compatibility with RFC 2560 [RFC2560], it is not prohibited to issue a certificate for an Authorized Responder using a different issuing key than the key used to issue the certificate being checked for revocation. However, such a practice is strongly discouraged, since clients are not required to recognize a responder with such a certificate as an Authorized Responder. „

Hier wird von dem derzeit benutzten Verfahren abgeraten. Dadurch muss der OCSP-Server auf eine der beiden anderen Varianten umgestellt werden.

CA-Zertifikat signiert OCSP-Antwort

Bei diesem Verfahren wird die OCSP-Antwort von dem CA-Zertifikat erstellt, das auch die Zertifikate ausgestellt hat. Auf der Basis der Komponentenaufteilung aus Kapitel 4.1 und deren Entscheidungskriterien wäre der Einsatz des CA-Zertifikats zum Signieren der OCSP-Antwort eine ungünstige Variante. Eines der Ziele in der Komponentenaufteilung war, dass externe Anfragen nicht die Ausstellung von Zertifikaten bzw. von Sperrlisten beeinflussen. Indem mit dem gleichen privaten Schlüssel signiert wird, kann eine Denial-of-Service-Attacke (DoS-Attacke) durch einen Angreifer durchgeführt werden.

Ein Lösungsansatz zum Verhindern der DoS-Attacke ist der Einsatz eines eigenen Hardware-Security-Moduls für den OCSP-Server. Dies setzt voraus, dass der Schlüssel aus dem bestehenden Hardware-Security-Modul exportiert werden kann bzw. exportiert werden darf.

Ein weiterer Lösungsansatz ist das Cachen bzw. Vorgenerieren von häufigen OCSP-Antworten. Diese Varianten sind im RFC 5019 (s. [AD07]) und im RFC 6960 (s. [MM13,

[16] http://tools.ietf.org/html/rfc6960

Kapitel 2.5 Response Pre-Production]) beschrieben, gleichzeitig werden jedoch auch die Sicherheitsbedenken im Hinblick auf die DoS-Attacken erwähnt (vgl. [AD07, Kapitel 7.4 Denial-of-Service Attacks] und [MM13, Kapitel 5.1.3. Denial-of-Service Attack]). Die Erkenntnis aus diesen Dokumenten ist, dass sowohl das Cachen als auch das Vorgenerieren der Antworten eine geringere Last bei OCSP-Anfragen bringen, jedoch bei DoS-Angriffen kein Unterschied zu erwarten ist.

Erstellen eines eigenen OCSP-Signers durch jedes CA-Zertifikat

Die Variante mit speziellen OCSP-Signer-Zertifikaten für jedes CA-Zertifikat hat den Vorteil, dass DoS-Attacken nicht die Zertifikatsausstellung und Sperrlistenerstellung beeinflussen. Demgegenüber steht der erhöhte Verwaltungsaufwand für die Ausstellung und Verwaltung der einzelnen OCSP-Signer-Zertifikate.

Für diese Probleme wäre es ein Lösungsansatz, diese Aufgabe automatisiert durch die CA-Software zu erledigen. Die Software muss erkennen, wann ein neues OCSP-Signer-Zertifikat benötigt wird, und dieses rechtzeitig ausstellen. Ebenso fallen die Verwaltung der einzelnen OCSP-Signer-Zertifikate und deren Zuordnung zur jeweiligen CA-Instanz in das Aufgabengebiet der CA-Software. Zusätzlich müssen diese Zertifikate veröffentlicht und an alle Entwickler weitergeleitet werden. Das Problem kann laut [MM13, Kapitel 4.2.2.2.1. Revocation Checking of an Authorized Responder] umgangen werden, indem in das OCSP-Signer-Zertifikat die Erweiterung „id-pkix-ocsp-nocheck“ aufgenommen wird. Dadurch wird der Client angewiesen, das OCSP-Signer-Zertifikat nicht zu prüfen. Diese Variante hat jedoch folgenden Nachteil:

> „A CA may specify that an OCSP client can trust a responder for the lifetime of the responder's certificate. The CA does so by including the extension id-pkix-ocsp-nocheck. This SHOULD be a non-critical extension. The value of the extension SHALL be NULL. CAs issuing such a certificate should realize that a compromise of the responder's key is as serious as the compromise of a CA key used to sign CRLs, at least for the validity period of this certificate. CAs may choose to issue this type of certificate with a very short lifetime and renew it frequently.“

Durch das Hinzufügen dieser Zertifikatserweiterung wird eine Kompromittierung des OCSP-Signer-Zertifikats mit der des CA-Zertifikats gleichgesetzt. Es wird empfohlen, die Gültigkeit dieses Zertifikats auf eine kurze Zeit zu beschränken.

5 Softwareentwicklung

In den verbreiteten Kryptographiebibliotheken besteht eine gute Unterstützung der clientseitigen Protokolle und Funktionen für die Prüfung von Zertifikaten, CRLs und OCSP-Abfragen. Wie ausgeprägt die Unterstützung der serverseitigen Funktionen vorhanden ist, muss erst in Erfahrung gebracht werden. Dieses Kapitel soll zeigen, inwieweit diese Server-Funktionen für die Implementierung eines CA-Software-Systems ausreichend sind. Bei den nachfolgenden Untersuchungen werden vor allem die folgenden Funktionen getestet:

1. Erstellung von X509-Zertifikaten nach RFC5280 (s. [DC08]) bzw. die Zertifikatsvorlagen aus Anhang A.
   Wie bereits in der Analyse der bestehenden CA-Systeme in Kapitel 2.3, werden auch bei der Untersuchung der Kryptographiebibliotheken die selben Maßstäbe verwendet. Die Zertifikatsvorlagen beinhalten alle derzeit benötigten Zertifikatserweiterung und stellen so die Minimalfunktionen für die jeweiligen Bibliotheken dar.
2. Erstellen einer CRL mit jeweils einem Zertifikat im folgenden Status:

   - gesperrt
   - widerrufen mit Grund keyCompromise
   - widerrufen mit Grund superseded
   - widerrufen mit Grund cessationOfOperation

   Die aufgelisteten Sperrlisteneinträge beinhalten die derzeit möglichen Einträge. Jede dieser Varianten muss mit der Kryptographiebibliotheken abbildbar sein.
3. OCSP

   - Parsen einer OCSP-Anfrage
   - Erstellen einer OCSP-Antwort

Diese drei Punkte beinhalten die wichtigsten kryptographischen Funktionen, welche von einem CA-Server System abgebildet werden müssen. Sie entsprechen den Aufgaben der Zertifikatserstellung (Punkt 1.) und der Abfrage der Widerrufsinformationen über Sperrlisten (Punkt 2.) und OCSP (Punkt 3.).

Verifikation der Testergebnisse

Die Verifikation der erstellten Zertifikate wird durch mehrere Vergleiche durchgeführt. Sie werden mit dem Windows-Zertifikatsviewer geöffnet und die Felder mit den Vorlagenzertifikaten verglichen. Dabei werden Feldinhalte und Signatur geprüft. Zudem

werden die Zertifikate mit dem Programm `Dumpasn1.exe` geparst und mit den geparsten Werten der Vorlagenzertifikate verglichen. Bei diesem Test werden die Feldinhalte auf der Ebene ihrer ASN1-Struktur überprüft. Ein weiterer Test erfolgt mit Hilfe des OpenSSL-Kommandozeilenprogramms; mit dem Befehl aus Listing 5.1 wird das Zertifikat geladen und als Text ausgegeben.

```
1 openssl.exe x509 -inform DER -in dienstcert.cer -text
```

Listing 5.1: OpenSSL-Überprüfung der Testzertifikate

Die erstellte Test-Sperrliste wird, wie die Zertifikate mit dem Windows-Sperrlistenviewer geöffnet. Dieser zeigt die Feldinhalte, beinhaltete gesperrte bzw. widerrufene Zertifikate und überprüft die Signatur der Sperrliste. Zusätzlich wird über ein OpenSSL-Kommandozeilenprogramm die Sperrliste geparst und als Text ausgegeben, dazu wird der Befehl aus Listing 5.2 verwendet.

```
1 openssl.exe crl -inform DER -in test.crl -text
```

Listing 5.2: OpenSSL-Überprüfung der Sperrliste

Die OCSP-Antwort wird über ein OpenSSL-Kommandozeilen-Programm geprüft. Dazu wird der Befehl in Listing 5.3 verwendet, und die Ausgabe des Programms zeigt die gesamte OCSP-Antwort, im Speziellen den Status des abgefragten Zertifikats.

```
1 openssl.exe ocsp -respin OcspResponse.bin -no_cert_verify -resp_text
```

Listing 5.3: OpenSSL-Überprüfung der OCSP-Antwort

5.1 OpenSSL

OpenSSL wurde bereits in Kapitel 2.3.5 beschrieben, wobei der Schwerpunkt auf den Kommandozeilenbefehlen zur Erstellung von Testzertifikaten lag. OpenSSL bietet gleichzeitig eine Bibliothek, die von Softwareentwicklern verwendet werden kann, um Zertifikate zu erstellen. Wie angesichts der Analyse des genannten Kapitels zu erwarten war, ist es auch mit der Bibliothek möglich, alle geforderten Testzertifikate zu erstellen. Die Einschränkungen hinsichtlich der Subject-Felder im Zertifikat bestehen bei der Verwendung der Programmierschnittstelle nicht. Wie in Listing 5.4 zu sehen ist, können die geforderten Subject-Felder gesetzt werden. Einige Felder werden mit deren Abkürzung, die in der Analyse problematischen Felder jedoch nur mit einem OpenSSL-spezifischen Kennzeichen (z.B. `NID_serialNumber`) erkannt.

```
X509_NAME *name = X509_get_subject_name(cert);
X509_NAME_add_entry_by_txt(name, "C", MBSTRING_UTF8, "AT",-1,-1,0);
X509_NAME_add_entry_by_txt(name, "CN", MBSTRING_UTF8,"Patrick Hagelkruys",-1,-1,0);
X509_NAME_add_entry_by_txt(name, "SN", MBSTRING_UTF8,"Hagelkruys",-1,-1,0);
X509_NAME_add_entry_by_NID(name, NID_givenName, MBSTRING_UTF8, "Patrick",-1,-1,0);
X509_NAME_add_entry_by_NID(name, NID_title, MBSTRING_UTF8, "Ing",-1,-1,0);
X509_NAME_add_entry_by_NID(name, NID_serialNumber, MBSTRING_ASC, "369108117853"
    ,-1,-1,0);
```

Listing 5.4: OpenSSL-Subject-Felder

Die Zertifikatserweiterungen in den Testzertifikaten wurden über die Programmierschnittstellen ohne Verwendung der OpenSSL-Konfigurationsdateien eingefügt. Einfache Erweiterungen werden ebenfalls über OpenSSL Identifier (vgl. Listing 5.5 Identifier Zeile 4, `NID_key_usage`) hinzugefügt. Die übergebenen Werte ähneln denen der Konfigurationsdateien.

```
X509V3_CTX ctx;
X509V3_set_ctx_nodb(&ctx);
X509V3_set_ctx(&ctx, cert, cert, NULL, NULL, 0);
X509_EXTENSION *ex = X509V3_EXT_conf_nid(NULL, &ctx, NID_key_usage, "critical,
    digitalSignature,keyEncipherment,dataEncipherment");
X509_add_ext(cert, ex, -1);
X509_EXTENSION_free(ex);
```

Listing 5.5: OpenSSL-Zertifikatserweiterung keyUsage

Zertifikatserweiterungen, die aus mehr als einem Wert bestehen, müssen über eigene Strukturen angegeben werden. Dazu gehören vor allem Erweiterungen, für die in den Konfigurationsdateien eigene Sections notwendig sind. Speziell die Zertifikatserweiterungen für das QC-Statement und die Dienstleistereigenschaft müssen über ihre ASN1-Struktur aufgebaut werden. Die einfache Erweiterung Dienstleistereigenschaft ist in Listing 5.6 dargestellt.

```
// Inhalt der Zertifikatserweiterung
ASN1_NULL *n = ASN1_NULL_new();
int lenNull = i2d_ASN1_NULL(n, NULL);
void *seqNull = malloc(lenNull);
unsigned char *p = (unsigned char*)seqNull;
i2d_ASN1_NULL(n, &p);
ASN1_NULL_free(n);

// Zertifikatserweiterungen werden immer als Octetstring gespeichert
ASN1_OCTET_STRING *os = ASN1_OCTET_STRING_new();
ASN1_OCTET_STRING_set(os, (unsigned char*)seqNull, lenNull);

// OID fuer Dienstleister generieren
int nid = OBJ_create("1.2.40.0.10.1.1.2", "Dienstleister", "Dienstleister");
X509_EXTENSION *ex = X509_EXTENSION_create_by_NID(NULL, nid, 0, os);
X509_add_ext(cert, ex, -1);

// Speicher freigeben (ex,os)
```

Listing 5.6: OpenSSL-Zertifikatserweiterung Dienstleistereigenschaft

Die Dokumentation für die OpenSSL-Bibliotheken ist nur rudimentär vorhanden. Die

meisten Informationen zum richtigen Umgang mit der Programmierschnittstelle finden sich in dem mitgelieferten Beispielcode bzw. in den Hilfsforen und Diskussionsgruppen. Da der OpenSSL-Quellcode öffentlich verfügbar ist, kann auch er zur Analyse von Problemen und zur Schnittstelleninterpretation herangezogen werden.

Das entwickelte Testprogramm befindet sich auf der beigelegten CD-ROM im Ordner `/Softwareentwicklung/OpenSSL/` (s. Anhang H)

5.2 Bouncycastle (Java und C#)

Bouncycastle ist eine Open-Source-Kryptographiebibliothek, die für Java und C# entwickelt wurde. Die offizielle Webseite des Projekts ist unter `http://www.bouncycastle.org/`[17] erreichbar. Diese Bibliothek wird in der CA-Software SimpleAuthority (s. Kapitel 2.3.4) verwendet, bei der einige Probleme bei den Zertifikatserweiterungen bestehen.

Im Gegensatz zu den Problemen bei der Analyse der CA-Software ist mit den Bouncycastle-Bibliotheken die Erstellung aller Testzertifikate möglich, sowohl mit der Java als auch mit der C#-Version. Bouncycastle arbeitet mit Generatoren, die mit den Grunddaten der jeweiligen Strukturen befüllt werden (vgl. Listing 5.7). Zertifikatserweiterungen werden zuerst in einer Hashtable gespeichert und ebenfalls dem Generator übergeben (vgl. Listing 5.7 Zeile 18-21). Anschließend erzeugt der Generator die TBS-Struktur[18], die im weiteren Verlauf signiert und zur abschließenden Erzeugung des Zertifikats verwendet wird.

```
1  V3TBSCertificateGenerator gen = new V3TBSCertificateGenerator();
2  ASN1Integer serial = new ASN1Integer(1);
3  gen.setSerialNumber(serial);
4  gen.setStartDate(new Time(ValidFrom.getTime()));
5  gen.setEndDate(new Time(ValidTo.getTime()));
6  String sSubject = "C=AT,O=A-Trust,OU=A-Trust,CN=www.a-trust.at";
7  X509Name subject = new X509Name(sSubject);
8  gen.setSubject(subject);
9  gen.setIssuer(caCert.getSubject());
10 gen.setSignature(
11         new AlgorithmIdentifier(
12                 PKCSObjectIdentifiers.sha1WithRSAEncryption,
13                 new DERNull())
14         );
15 subPubKeyInfo = SubjectPublicKeyInfoFactory.createSubjectPublicKeyInfo(pubKey);
16 gen.setSubjectPublicKeyInfo(subPubKeyInfo);
17
18 Hashtable extensions
19 // Zertifikatserweiterungen erstellen
20 X509Extensions  ex = new X509Extensions(extensions);
21 gen.setExtensions(ex);
22
23 TBSCertificate tbs = gen.generateTBSCertificate();
```

Listing 5.7: Bouncycastle-Zertifikatsgenerator (Java)

[17] `http://www.bouncycastle.org/sharp/` für die C#-Version
[18] TBS = to be signed

Für alle benötigten Standard-Zertifikatserweiterungen wird von der Bibliothek eine Struktur zur Verfügung gestellt. Nur die Erweiterungen Dienstleistereigenschaft und QC-Statement müssen über ihre ASN1-Struktur zusammengestellt werden; dieser Vorgang ist im Listing 5.8 für das QC-Statement dargestellt.

```
1 Asn1EncodableVector vec = new Asn1EncodableVector();
2 vec.Add(new DerSequence(new DerObjectIdentifier("0.4.0.1862.1.1")));
3 vec.Add(new DerSequence(new DerObjectIdentifier("1.3.6.1.5.5.7.11.1")));
4 Asn1Sequence seq = new DerSequence(vec);
5 DerOctetString content = new DerOctetString(seq.ToAsn1Object());
6 X509Extension ext = new X509Extension(true, content);
7 ExtensionList.Add(new DerObjectIdentifier("1.3.6.1.5.5.7.1.3"), ext);
```

Listing 5.8: Bouncycastle-Zertifikatserweiterung QC-Statement (C#)

Eine Dokumentation ist für beide Bibliotheken vorhanden. Sie enthält eine Übersicht über die Klassen mit Methoden und Eigenschaften. Zusätzlich werden einfache Beispiele für die Verwendung der API angeboten. Da der Quellcode der Bibliotheken verfügbar ist, können er und die darin enthaltenen UnitTest-Programme zum besseren Verständnis der Funktionsaufrufe verwendet werden. Bouncycastle ist in der Java Community weit verbreitet, für viele Probleme lässt sich daher schnell Hilfe finden. Durch die starke Ähnlichkeit der Java- und der C#-Programmierschnittstellen können die Einträge in Hilfsforen und Diskussionsgruppen jeweils für beide Sprachen verwendet werden.

Die entwickelten Testprogramme befinden sich auf der beigelegten CD-ROM im Ordner `/Softwareentwicklung/BouncycastleJava/` und `/Softwareentwicklung/BouncycastleCSharp/` (s. Anhang H)

5.3 Microsoft Cryptographic Application Programming Interface (CryptoAPI)

Das Microsoft Cryptographic Application Programming Interface (CryptoAPI) ist eine Programmierschnittstelle zur Nutzung der im Windows-Betriebssystem integrierten Kryptographiefunktionen. Diese Funktionen werden zumeist durch Microsoft-Programme und Betriebssysteme verwendet. Die bekannteste und häufigste Anwendung ist vermutlich der Microsoft Internet Explorer.

Die Erstellung der Testzertifikate ist ohne Einschränkung möglich. Für die Dienstleister-Erweiterung und das QC-Statement müssen die ASN1-Strukturen händisch zusammengestellt werden. Dieser Vorgang ist für die Dienstleistereigenschaft noch recht simpel (vgl. Listing 5.9), für das QC-Statement hingegen kompliziert und unübersichtlich (vgl. Listing 5.10).

```
1 BYTE asn1Null[] = { 0x05, 0x00 };
2 extension.fCritical = false;
3 extension.pszObjId = "1.2.40.0.10.1.1.2";
4 extension.Value.cbData = sizeof(asn1Null);
5 extension.Value.pbData = (BYTE*)&asn1Null;
```

Listing 5.9: MS-CryptoApi-Zertifikatserweiterung Dienstleistereigenschaft

Im Listing 5.10 Zeile 1-10 wird der erste Object Identifier für das QC-Statement codiert. In Zeile 14-19 wird der generierte Object Identifier in eine ASN1-Sequence verpackt, die abschließend in die Zertifikatserweiterungsstruktur codiert wird (vgl. Zeile 23-29)

```
1  LPCSTR oid1 = "0.4.0.1862.1.1";
2  CRYPT_DER_BLOB seqBlob1;
3  CryptEncodeObject(X509_ASN_ENCODING, X509_OBJECT_IDENTIFIER, &oid1,
4          NULL, &seqBlob1.cbData);
5  seqBlob1.pbData = (BYTE*)CryptMemAlloc(seqBlob1.cbData);
6  CryptEncodeObject(X509_ASN_ENCODING, X509_OBJECT_IDENTIFIER, &oid1,
7          seqBlob1.pbData, &seqBlob1.cbData);
8  CRYPT_SEQUENCE_OF_ANY seqData1;
9  seqData1.cValue = 1;
10 seqData1.rgValue = (CRYPT_DER_BLOB*)&seqBlob1;
11
12 /* ... selber Block fuer zweite OID ... */
13
14 CRYPT_DER_BLOB seqBlob[2];
15 CryptEncodeObject(X509_ASN_ENCODING, X509_SEQUENCE_OF_ANY, &seqData1,
16         NULL, &seqBlob[0].cbData);
17 seqBlob[0].pbData = (BYTE*)CryptMemAlloc(seqBlob[0].cbData);
18 CryptEncodeObject(X509_ASN_ENCODING, X509_SEQUENCE_OF_ANY, &seqData1,
19         seqBlob[0].pbData, &seqBlob[0].cbData);
20
21 /* ... selber Block fuer zweite OID ... */
22
23 extension.fCritical = true;
24 extension.pszObjId = "1.3.6.1.5.5.7.1.3";
25 CryptEncodeObject(X509_ASN_ENCODING, X509_SEQUENCE_OF_ANY, &seqData,
26         NULL, &extension.Value.cbData);
27 extension.Value.pbData = (BYTE*)CryptMemAlloc(extension.Value.cbData);
28 CryptEncodeObject(X509_ASN_ENCODING, X509_SEQUENCE_OF_ANY, &seqData,
29         extension.Value.pbData, &extension.Value.cbData);
```

Listing 5.10: MS-CryptoApi-Zertifikatserweiterung QC-Statement

Die Erstellung der Sperrliste funktioniert ähnlich wie die Erstellung eines Zertifikats. Zuerst wird eine Grundstruktur initialisiert und mit den vorgesehen Werten befüllt, zusätzlich können noch Erweiterungen eingefügt werden. Anschließend wird diese Grundstruktur an die `CryptSignAndEncodeCertificate`-Funktion übergeben, welche die signierte und codierte Sperrliste bzw. das Zertifikat zurückliefert. Die Erstellung der OCSP-Antwort funktioniert ähnlich, jedoch müssen hier einige ineinander verschachtelte Strukturen codiert werden, wodurch der Ablauf komplexer wird.

Die Dokumentation der Befehle ist im Microsoft Developer Network[19] aufgelistet. Trotz der Cryptography Reference (s. [Mic14a]) und einiger Beispielprogramme ist die Erstel-

[19] `http://msdn.microsoft.com/`

lung von Zertifikaten, Sperrlisten und OCSP-Antworten wenig bis gar nicht beschrieben. Für diesen Anwendungsfall ist auch in den zahlreichen Microsoft-Foren kaum Hilfe zu finden. Da für diese Bibliotheken auch kein Quellcode verfügbar ist, kann nur anhand der Dokumentation vorgegangen werden.

Das entwickelte Testprogramm befindet sich auf der beigelegten CD-ROM im Ordner `/Softwareentwicklung/MsCryptoApi/` (s. Anhang H)

5.4 IAIK Cryptographic Service Provider (IAIK-JCE)

Der IAIK Cryptographic Service Provider für Java ist eine Programmierbibliothek für Kryptographiefunktionen, entwickelt von der Technischen Universität Graz. Die Beschreibung der Bibliothek lautet gemäß ihrer Homepage (vgl. [SIC14]) wie folgt:

> „The IAIK Provider for the Java™ Cryptography Extension (IAIK-JCE) is a set of APIs and implementations of cryptographic functionality, including hash functions, message authentication codes, symmetric, asymmetric, stream, and block encryption, key and certificate management. It supplements the security functionality of the default JDK.“

Die Bibliothek bietet einfache Funktionen für die Erstellung von Zertifikaten, Sperrlisten und OCSP-Antworten. Die Erstellung der Vorlagenzertifikate ist ohne Einschränkung möglich. Wie in Listing 5.11 zu sehen ist, wird die Erstellung des Zertifikatsgrundgerüstes über die Klasse `X509Certificate` ermöglicht. Der Vorteil der IAIK-Bibliotheken ist, dass die X509Certificate-Klasse (`iaik.x509.X509Certificate`) eine direkte Unterklasse der Java X509Certificate-Klasse (`java.security.cert.X509Certificate`) ist. Dadurch kann sie nach dem Signieren in Zeile 19 an bestehende Java-Funktionen übergeben werden, als wäre sie die Java-eigene Klasse.

```
1  X509Certificate cert = new X509Certificate();
2
3  cert.setSerialNumber(BigInteger.valueOf(0x1234L));
4  cert.setSubjectDN(subject);
5  cert.setPublicKey(ca_keys.getPublic());
6  cert.setIssuerDN(subject);
7
8
9  // ValidFrom
10 Calendar ValidFrom = Calendar.getInstance();
11 cert.setValidNotBefore(ValidFrom.getTime());
12
13 // ValidTo
14 Calendar ValidTo = Calendar.getInstance();
15 ValidTo.add(Calendar.YEAR, 5);
16 cert.setValidNotAfter(ValidTo.getTime());
17
18 // sign Certificate
19 cert.sign(AlgorithmID.sha1WithRSAEncryption, ca_keys.getPrivate());
```

Listing 5.11: IAIK-Grundgerüst

Die Erstellung von Zertifikatserweiterungen funktioniert auf der Basis der vorgefertigten Klassen für fast alle benötigten Erweiterungen, wie anhand des QC-Statements in Listing 5.12 Zeile 1-8 zu sehen ist. Unbekannte Erweiterungen wie die Dienstleistereigenschaft können über eine eigene Klasse `UnknownExtension` hinzugefügt werden, wie im Listing 5.12 in Zeile 10-13 dargestellt ist.

```
1  // Erweiterung QC-Statement (vorgefertigte Erweiterung)
2  QCStatement[] qcStatements = new QCStatement[2];
3  qcStatements[0] = new QCStatement(new ObjectID("0.4.0.1862.1.1"));
4  qcStatements[1] = new QCStatement(new ObjectID("1.3.6.1.5.5.7.11.1"));
5
6  QCStatements qcStatementsExt = new QCStatements(qcStatements);
7  qcStatementsExt.setCritical(true);
8  cert.addExtension(qcStatementsExt);
9
10 // Erweiterung Dienstleistereigenschaft (unbekannte Erweiterung)
11 UnknownExtension ext = new UnknownExtension(new ObjectID("1.2.40.0.10.1.1.2"));
12 ext.init(new NULL());
13 cert.addExtension(ext);
```

Listing 5.12: IAIK-Zertifikatserweiterungen

Die IAIK-Bibliotheken bieten eine gute Dokumentation mit Beispielen zum Einsatz der einzelnen Klassen an. Zusätzlich werden Beispielprogramme mitgeliefert, die für die Erstellung eigener Programme hilfreich sind. Die Bibliotheken werden von vielen Entwicklern benützt, daher sind Lösungsvorschläge für viele Probleme in Hilfsforen und Diskussionsgruppen zu finden.

Das entwickelte Testprogramm befindet sich auf der beigelegten CD-ROM im Ordner `/Softwareentwicklung/IAIK/` (s. Anhang H). Die benötigten IAIK-Bibliotheken dürfen aufgrund der Lizenzeinschränkungen nicht mitgeliefert werden, können aber unter `https://jce.iaik.tugraz.at/` bezogen werden.

5.5 Weitere Bibliotheken

Java Sun Security CertAndKeyGen

Das Java OpenJDK enthält Funktionen zur einfachen Erstellung von Zertifikaten. Die Beschreibung der entsprechenden Klasse ist unter [LLC14] zu finden:

> „Generate a pair of keys, and provide access to them. This class is provided primarily for ease of use. This provides some simple certificate management functionality. Specifically, it allows you to create self-signed X.509 certificates as well as PKCS 10 based certificate signing requests.“

Die Funktionen beschränken sich auf die Erstellung von selbst signierten Zertifikaten, es wird keine Unterstützung für Zertifikatsketten, Sperrlisten oder OCSP angeboten.

Zudem wird auf dieser Seite beschrieben, dass derzeit keine Zertifikatserweiterungen unterstützt werden.

Ein Testprogramm befindet sich auf der beigelegten CD-ROM im Ordner `/Softwareentwicklung/JavaSunSecurityCertAndKeyGen/` (s. Anhang H)

GNU Crypto

GNU Crypto ist eine Open-Source-Bibliothek für Java mit dem Ziel, beweisbar korrekte Implementierungen von kryptographischen Grundfunktionen und Werkzeugen bereitzustellen (vgl. [Fou06]). Wie der Funktionsübersicht[20] von GNU Crypto entnommen werden kann, wird von der Bibliothek keine Erstellung von Zertifikaten, Sperrlisten oder OCSP-Antworten unterstützt.

pyCrypto - The Python Cryptography Toolkit

pyCrypto ist eine Kryptographiebibliothek für die Programmiersprache Python. Die Read-Me-Datei der pyCrypto-Distribution (s. [Lit13]) sagt hierzu Folgendes:

> „This is a collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.). [...] This section is essentially complete, and the software interface will almost certainly not change in an incompatible way in the future [...]."

Der offizielle Webauftritt von pyCrypto ist unter `http://www.pycrypto.org/` zu finden. Wie aus der API-Dokumentation[21] zu erkennen ist, werden keine Funktionen zur Erstellung von Zertifikaten, Sperrlisten oder OCSP-Antworten bereitgestellt.

pyOpenSSL - Python interface to the OpenSSL library

pyOpenSSL ist ein Python-Wrapper-Modul für die OpenSSL-Bibliotheken. Die Webseite des Projekts ist unter `http://pyopenssl.sourceforge.net/` zu erreichen. In der Dokumentation [Cal11] der Bibliothek ist folgendes Abstract angegeben:

> „This module is a rather thin wrapper around (a subset of) the OpenSSL library. With thin wrapper I mean that a lot of the object methods do nothing more than calling a corresponding function in the OpenSSL library."

pyOpenSSL unterstützt die Erstellung von Zertifikaten und Sperrlisten, bietet jedoch keine Funktionen zum Erstellen von OCSP-Antworten. Die Testzertifikate nach den Vor-

[20] `http://www.gnu.org/software/gnu-crypto/algorithms.html`
[21] `https://www.dlitz.net/software/pycrypto/api/current/`

lagen können aufgrund der Einschränkungen der Programmierschnittstelle nicht erstellt werden. Es ist nicht möglich, die Zertifikatserweiterungen `Certificate Policies`, `CRL Distribution Points`, `Authority Information Access`, `qcStatement` oder die `Dienstleistereigenschaft` wie gefordert zu erstellen. Die Felder des Zertifikats-Subject sind beschränkt. Für die Testzertifikate fehlen die Felder `serialNumber`, `givenName`, `surName` und `title`.

Ein Testprogramm befindet sich auf der beigelegten CD-ROM im Ordner `/Softwareentwicklung/pyOpenSSL/` (s. Anhang H)

M2Crypto: A Python crypto and SSL toolkit

M2Crypto ist ein Python-Wrapper-Modul für die OpenSSL-Biblotheken. Die offizielle Webseite des Projekts findet sich unter `http://pyopenssl.sourceforge.net/`[22]. Der Beschreibung der Homepage [Mac13] des Projekts ist Folgendes zu entnehmen:

> „M2Crypto is the most complete Python wrapper for OpenSSL featuring RSA, DSA, DH, HMACs, message digests, symmetric ciphers (including AES); SSL functionality to implement clients and servers; HTTPS extensions to Python's httplib, urllib, and xmlrpclib; unforgeable HMAC'ing AuthCookies for web session management; FTP/TLS client and server; S/MIME; ZServerSSL: A HTTPS server for Zope and ZSmime: An S/MIME messenger for Zope. M2Crypto can also be used to provide SSL for Twisted."

Sowohl von der Homepage als auch von der mit der Software mitgelieferten ReadMe-Datei (vgl. [Mac13] und [Toi13]) ist zu erkennen, dass die Software keine Erstellung von Zertifikaten, Sperrlisten oder OCSP-Anfragen unterstützt.

5.6 Zusammenfassung

Wie in Tabelle 5.1 zu ersehen ist, werden von den vier Bibliotheken OpenSSL, Bouncycastle, Microsoft CryptoAPI und IAIK alle benötigten Funktionen zur Verfügung gestellt. Bei den beiden C/C++-basierten Bibliotheken OpenSSL und Microsoft CryptoAPI ist die Erstellung der nicht integrierten Erweiterungen ein aufwendiger Prozess. Vermutlich durch die Hostsprache bzw. die C-kompatible Schnittstelle wird oftmals dynamisch Speicher alloziert, für den der richtige Zeitpunkt der Freigabe nicht immer ohne Weiteres zu erkennen ist. In Bezug auf die Dokumentation lieferten sowohl die Bouncycastle- als auch die IAIK-Bibliotheken sehr gute Hilfsseiten und Beispielprogramme mit.

[22] Letzte verfügbare Kopie s. WayBack Machine `https://web.archive.org/web/20130825211205/http://chandlerproject.org/Projects/MeTooCrypto`

Bibliothek	**Programmiersprache**	**CA-Zertifikat**	**SAN**[1]	**UPN**[2]	**QC-Statement**[3]	**Dienstleister**[4]	**Testsperrliste**	**OCSP**	**Dokumentation**	**Community Support**	**Open Source**
OpenSSL	C/C++	✓	✓	✓	✓	✓	✓	✓	◯	++	✓
Bouncycastle	C#	✓	✓	✓	✓	✓	✓	✓	+	+	✓
Bouncycastle	Java	✓	✓	✓	✓	✓	✓	✓	++	++	✓
Microsoft Crypto API	C/C++	✓	✓	✓	✓	✓	✓	✓	◯	−−	
IAIK	Java	✓	✓	✓	✓	✓	✓	✓	++	◯	
GNU Crypto	Java										✓
Java Sun Security CertAndKeyGen	Java	✓							++	◯	
PyCrypto	Python										✓
pyOpenSSL	Python	✓					✓		◯	◯	✓
M2Crypto	Python										✓

++ sehr gut + gut ◯ zufriedenstellend − schlecht −− sehr schlecht

[1] Webserverzertifikat mit mehreren Domains im Subject Alternative Name (SAN)
[2] Zertifikat mit UPN
[3] Zertifikat mit Qualified Certificate Statement (QC)
[4] Webserverzertifikat mit Dienstleistereigenschaft(OID)

Tabelle 5.1: Zusammenfassung der Ergebnisse der Softwareentwicklung

Die weiteren getesteten Bibliotheken unterstützten teilweise keine der benötigten Funktionen bzw. nicht im ausreichenden Umfang. Bei den beiden OpenSSL-Wrapper-Bibliotheken PyCrypto und M2Crypto wurde der Fokus der Entwicklung auf die Kryptographiefunktionen gelegt. M2Crypto unterstützt keine Möglichkeiten der Zertifikatserstellung, PyCrypto hingegen liefert nur eine rudimentäre Funktionalität.

6 Testszenarien

Dieses Kapitel beschreibt Testszenarien, um die Stabilität und Korrektheit des CA-Server-Systems zu prüfen. Die nachfolgenden Kapitel legen zusätzliche Möglichkeiten der Überprüfung des CA-Systems dar, ersetzen jedoch nicht den Vorgang der Aktivierung und Kontrolle einzelner Testfälle.

6.1 Alle Zertifikate erneut ausstellen

Das derzeit verwendete CA-Server-System stellt bereits seit über zehn Jahren Zertifikate aus, die als Referenzwerte für das neue System herangezogen werden können. Alle für die Zertifikatsausstellung relevanten Daten müssen archiviert und aufbewahrt werden. Daher können die Eingabedaten der Zertifikatsausstellung rekonstruiert werden.

Durch ein zu entwickelndes Programm werden alle bereits ausgestellten Zertifikate mit dem neuen CA-Server-System nochmals erstellt. Somit ist ein direkter Vergleich der Ergebnisse des alten Systems mit denen des neuen möglich. Bei diesem Vergleich der Zertifikate sind jedoch einige Punkte zu beachten. Die Zertifikatsseriennummern müssen von den Vergleichen ausgeschlossen werden, da sie vom CA-System entweder zufällig oder fortlaufend vergeben werden und daher nicht unbedingt mit den Werten der alten Zertifikate übereinstimmen. Für die einfachere Zertifikatsausstellung ist es von Vorteil, für jede Ausstellung den gleichen Schlüssel zu verwenden. In diesem Fall muss dieser ebenfalls aus den Vergleichen entfernt werden. Für die Felder im Grundgerüst des Zertifikats (vgl. 2.1.3) ist die Reihenfolge vorgegeben, nicht aber für die Zertifikatserweiterungen bzw. in einigen Fällen für deren Unterstruktur (z.B.: `Subject Alternative Name`). Ebenso trifft das Problem der unterschiedlichen Reihenfolge auf die Werte innerhalb des Zertifikats-Subjects zu. Daher muss beim Vergleich dieser Werte die Reihenfolge ignoriert werden.

Aufgrund dieser Schwierigkeiten und der Anzahl der zu kontrollierenden Zertifikate ist es sinnvoll, für diesen Vorgang ein Programm zu entwickeln, das die aufgezählten Punkte berücksichtigt. Für die zukünftige Wartung und Weiterentwicklung ist es empfehlenswert, eine repräsentative Menge der Testdaten zu extrahieren, um so ein kleineres Testumfeld zu generieren. Dieses wird anschließend für Abnahmetests der neuen Version verwendet.

Wie in [Sab14] beschrieben, ist ein Ziel dieser Tests die Kontrolle, ob alle Zertifikatserweiterungen, die derzeit von der CA-Software unterstützt werden, auch im neuen System verwendet werden können.

6.2 Automatisierung der Client-Software

Neue Zertifikate werden zumeist über die bestehende Registrierungssoftware ausgestellt. Bei diesem Prozess werden die Daten des Kunden über eine Maske erfasst und anschließend mit einer RO-Karte bestätigt. Der nachfolgende Prozess der Zertifikatserstellung und der Transfer der Zertifikate auf die Smartcard des Kunden wird über das der CA vorgelagerte System und die Registrierungssoftware durchgeführt.

Durch GUI-Automatisierungsprogramme wie z.B. AutoIt[23], Automa[24] oder das Robot Framework[25] ist es möglich, die Registrierungssoftware so zu steuern, dass aufeinanderfolgende Zertifikate ausgestellt werden. Zu beachten ist, dass zwischen den einzelnen Aktivierungen die Smartcard des Kunden zurückgesetzt wird bzw. mehrmals aktivierbar sein muss. Für diesen Testfall ist es nicht von Bedeutung, dass der Zertifikatsinhalt gleichbleibend ist.

Ziel ist es, die Zusammenarbeit des neuen CA-Systems mit dem vorgelagerten System zu testen. Zudem wird das Verhalten der beiden Systeme unter Last getestet.

6.3 Test der parallelen Verarbeitung und des Lastverhaltens

Bei der im letzten Kapitel beschriebenen Automatisierung der Client-Software werden einige Vorgänge nur zwischen dem vorgelagerten System und der Registrierungssoftware durchgeführt, wobei die CA-Software nicht involviert ist. Der nachfolgend beschriebene Test soll die parallele Verarbeitung mehrerer gleichzeitiger Anfragen der CA-Software testen. Entsprechend muss das vorgelagerte System simuliert werden, um in kurzer Zeit viele Anfragen an die CA-Software zu senden.

Der Test erhält eine höhere Wirksamkeit, wenn mehrere parallele Prozesse bzw. Computer gleichzeitig Anfragen senden. Das Hauptaugenmerk liegt auf dem Verhalten des Gesamtsystems unter hoher Last, wobei auf Speicherauslastung und Reaktionsgeschwindigkeit zu achten ist. Wichtig ist auch die Erholungsrate des Systems nach dem Beenden des Tests.

Dieser Testfall lässt auch erkennen, wie die in Kapitel 4.1 erarbeitete Lösung zur Komponentenaufteilung Probleme mit hoher Last bewältigt. Ziel der Komponentenaufteilung war, dass eine große Anzahl an eingehenden Anfragen nicht die Sperrlistenerstellung beeinflussen darf. Daher sollte gleichzeitig mit diesem Test die Sperrlistenerstellung

[23] http://www.autoitscript.com/
[24] http://www.getautoma.com/
[25] http://robotframework.org/

bzw. die Verzögerung der Sperrlistenerstellung zum erwarteten Zeitpunkt kontrolliert werden.

6.4 Fuzzing

Fuzzing ist eine effiziente Methode zum Aufspüren von Softwarebugs. Bei dieser Technik werden fehlerhafte Daten gebildet, die der zu testenden Anwendung als Eingabedaten übergeben werden. Justin Seitz [Sei09] unterscheidet zwischen zwei Arten von Fuzzing-Programmen (kurz Fuzzer). Einerseits gibt es generierende Fuzzer, bei denen zufällige Daten erzeugt werden, andererseits finden sich mutierende Fuzzer, bei denen gültige Requests verändert werden.

Fuzzing kann zum Testen der externen Schnittstelle der CA-Software eingesetzt werden. Für das nachfolgende Beispiel eines Fuzzing-Tools wird der OCSP-Server als Testziel gewählt. Dieser Server wird über HTTP-POST- oder HTTP-GET-Requests angesprochen, als Parameter werden Binärdaten übergeben. Zur Erstellung der Testprogramme wird Python verwendet, während als Basis für die weitere Entwicklung das in Listing 6.1 dargestellte Programm dient.

```
1  # Lesen des OCSP-Requests
2  f = open("in/OcspReq.bin","rb")
3  data = f.read()
4  f.close()
5
6  # Senden des OCSP-Requests an den Server
7  headers = { "content-type" : "application/ocsp-request",
8                          "content-length" : 0}
9  headers["content-length"] = len(data)
10 req = urllib2.Request("http://...", data, headers)
11 resp = urllib2.urlopen(req)
12 responseData = resp.read()
```

Listing 6.1: Fuzzing-Basis-Script

In Zeile 1-4 wird eine gültige OCSP-Anfrage eingelesen, anschließend wird in Zeile 7-12 diese Anfrage an den OCSP-Server übergeben und die Antwort des Servers gelesen.

Ein generierender Fuzzer erzeugt zufällige Binärdaten. Daher wird über das Pseudo-Random-Number-Modul von Python ein zufälliger Request erzeugt und an den Server gesendet. Dieser Ablauf ist in Listing 6.2 dargestellt.

```
1 # Erzeugen eines zufaelligen Requests
2 count = random.randint(50,500)
3 data = os.urandom(count)
4
5 # Senden des OCSP-Requests an den Server
6 # ... OCSP Request = data
```

Listing 6.2: Generierender Fuzzer

In Zeile 2 wird eine zufällige Größe der zu erzeugenden OCSP-Anfrage erstellt; für dieses Beispielprogramm wird eine Begrenzung zwischen 50 Bytes und 500 Bytes gewählt. In Zeile 3 werden zufällige Binärdaten in der vorher gewählten Größe erstellt. Dadurch wird für jeden Durchlauf eine beliebig große, mit zufälligen Daten befüllte Anfrage erzeugt.

Ein mutierender Fuzzer modifiziert eine bestehende Anfrage zufällig. Das Beispielprogramm in Listing 6.3 ersetzt einen Block von Binärdaten aus dem bestehende Request mit einem unterschiedlich großen Block aus Zufallsdaten.

```
1  # Lesen des OCSP-Requests
2  f = open("in/OcspReq.bin","rb")
3  reqData = f.read()
4  f.close()
5
6  # Modifizieren des OCSP-Requests
7  pos = random.randint(0,len(reqData)-1)
8  removeLen = random.randint(1,len(reqData)-pos)
9  addLen = random.randint(1,500)
10 newData = os.urandom(addLen)
11 modData = reqData[0:pos] + newData + reqData[pos + removeLen:]
12
13 # Senden des OCSP-Requests an den Server
14 # ... OCSP Request = modData
```

Listing 6.3: Mutierender Fuzzer

In Zeile 1-4 wird die bestehende OCSP-Anfrage geladen, die in den nachfolgenden Zeilen 7-11 modifiziert wird. In Zeile 7 wird die Position des zu ersetzenden Blocks zufällig erzeugt, in der darauf folgenden Zeile wird die Länge des Blocks ermittelt. Zeile 9 erzeugt die Länge des neuen Blocks, der in Zeile 10 erzeugt wird. Abschließend wird die neue Anfrage in Zeile 11 aus den bestehenden und den neuen Daten zusammengesetzt.

Die drei Testprogramme befinden sich auf der beigelegten CD-ROM im Ordner `/Softwaretests/Fuzzing/`[26] (s. Anhang H).

[26] Die URL des OCSP-Servers wurde aus den Programmen entfernt

7 Zusammenfassung der Ergebnisse und Ausblick

Die vorliegende Arbeit beschäftigt sich mit der Erstellung einer CA-Server-Software für den Einsatz bei der Firma A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH. Ziel ist die Ablösung des bestehenden Systems. Zu diesem Zweck wurden in Kapitel 2.3 dreizehn verschiedene CA-Systeme untersucht und verglichen. Nur zwei von ihnen sind in der Lage, die geforderten Testzertifikate zu erstellen. Mithilfe der Analyse konnten einige Probleme, die bei einer Eigenentwicklung auftreten, erkannt werden. Dies betrifft unter anderem die mangelnde Unterstützung der Zertifikats-Subject-Felder und die unterschiedlichen Ansätze, kompliziertere Zertifikatserweiterungen einzufügen. Deutlich wurde zudem die starke Verbreitung von OpenSSL als Kryptographiebibliothek in den untersuchten Programmen.

Nach einer Präzisierung der Anforderungen in Kapitel 3 und einer Auflistung der erkannten Design- und Implementierungsprobleme wurde in Kapitel 4 mit der Beschreibung der internen Abläufe innerhalb der CA-Software begonnen. Die Darstellung der Ergebnisse in Ablaufdiagrammen dient dem Verständnis und ist für die weitere Vorgehensweise sowie die Implementierung der CA-Software von Bedeutung. Vor allem die benötigte Interaktion der verschiedenen Aufgabengebiete bzw. Komponenten lässt sich durch die schematische Darstellung erkennen.

Eine Anforderung aus der Präzisierung der Aufgabenstellung war die Diskussion der Komponentenaufteilung unter Berücksichtigung der Verfügbarkeit und Stabilität. Dazu wurden in Kapitel 4.1 verschiedene Ansätze besprochen. Einige von ihnen basieren auf den Erkenntnissen aus der Analyse der CA-Software-Systeme in Kapitel 2.3. Die Zusammenfassung des Kapitels bietet eine tabellarische Darstellung der einzelnen Komponenten und ihrer Eigenschaften.

Um die Ausfallsicherheit zu erhöhen und eine Lastverteilung zu erreichen, wurden im anschließenden Kapitel 4 Varianten diskutiert. Die in den Anforderungen aufgelisteten Punkte zu Synchronisierungsproblemen, fortlaufenden Seriennummern und doppelter Erstellung von Sperrlisten wurden bearbeitet und im weiteren Vorgehen die benötigten Softwareanpassungen, Wartungsprobleme und Kosten dargestellt.

Kapitel 4.3 stellte die Zertifikatsformate und -regeln einiger analysierter CA-Software-Systeme dar. Zusätzlich wurden zwei weitere Konzepte für Zertifikatsregeln diskutiert. Die Zusammenfassung des Kapitels lieferte eine Übersicht über die bearbeiteten Formate mit den jeweils erarbeiteten Eigenschaften. Die in den Anforderungen beschriebenen Punkte zur Protokollierung und Nachvollziehbarkeit wurden in diesem Kapitel nur kurz bearbeitet, boten jedoch eine denkbare Lösung an.

Das nachfolgende Kapitel zum Thema Sperrlisten und deren Erstellung zeigte zwei Möglichkeiten zur Berechnung der Erstellungszeiten. Anschließend wurde begründet, welche der beiden Varianten das geeignete Vorgehen ist. Es wurden die möglichen Fälle der Sperrlistenerstellung anhand der entsprechenden Formel graphisch dargestellt. Das zweite Problem der Sperrlisten ist die Dateigröße; dazu wurden sechs Lösungsmöglichkeiten beschrieben. Zwei dieser Varianten wurden mit Beispielberechnungen untermauert, um ihre Effizienz darzustellen.

Das letzte Kapitel der Systemanalyse beschrieb die im RFC6960 angeführten Varianten der möglichen Signatoren für die OCSP-Antworten. Es wurden die Vorteile und Nachteile der beiden Varianten dargestellt, ohne dass eine der beiden Möglichkeiten als die optimale Lösung erkennbar war.

Kapitel 5 zur Softwareentwicklung untersuchte neun unterschiedliche Kryptographiebibliotheken. Mit ihnen wurden Testzertifikate erstellt, wie sie schon in der Analyse der CA-Software-Systeme Anwendung fanden. Aus diesen Bibliotheken können vier die geforderten Zertifikate erstellen; dies wurde mit Beispielprogrammen nachgewiesen, die auf der beigelegten CD-ROM enthalten sind (s. Anhang H)

Kapitel 6 zu den Softwaretests behandelte Verfahren zur Überprüfung des entwickelten CA-Software-Systems. Die angeführten Szenarien beinhalten Tests, die außerhalb der Standardfälle durchgeführt wurden. Im Abschnitt über das Fuzzing wurden zwei einfache Programme entwickelt, um aufzuzeigen, wie z.B. der OCSP-Server auf Stabilität getestet werden kann.

Ausblick

Für die weitere Vorgehensweise ist eine Entscheidung über die eingesetzten Kryptographiebibliotheken zu treffen. Die vier Produkte, die in Kapitel 5 die benötigten Anforderungen erfüllt haben, unterscheiden sich vor allem durch die eingesetzte Programmiersprache bzw. ihre Dokumentation und Community Support.

Ein wichtiger Punkt, der in der vorliegenden Arbeit nicht besprochen wurde, ist der Transfer der bestehenden Daten aus dem alten CA-System in die neue Eigenentwicklung. Dazu zählen auch die ca. 100 Zertifikatsformate, die derzeit für die Erstellung der Produkte eingesetzt werden. Die Korrektheit der transferierten Zertifikatsformate kann mit den in Kapitel 6.1 beschriebenen Tests überprüft werden.

Auf der Basis der gewählten Kryptographiebibliothek und Programmiersprache kann durch die mitgelieferten Programmbeispiele relativ schnell ein erster Prototyp der CA-Software entwickelt werden. Die benötigten Funktionen sind in den Beispielen bereits vorhanden.

Anhang

A ASN.1 Strukturen der Referenz-Zertifikate

A.1 Zertifikate mit mehreren Domainnamen im Subject Alternative Name

Die Domainnamen im Subject Alternative Name befinden sich in den Zeilen 124 bis 130.

```
  0 1444: SEQUENCE {
  4 1164:   SEQUENCE {
  8    3:     [0] {
 10    1:       INTEGER 2
       :       }
 13    3:     INTEGER 968907
 18   13:     SEQUENCE {
 20    9:       OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
 31    0:       NULL
       :       }
 33  135:     SEQUENCE {
 36   11:       SET {
 38    9:         SEQUENCE {
 40    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
 45    2:           PrintableString 'AT'
       :           }
       :         }
 49   72:       SET {
 51   70:         SEQUENCE {
 53    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
 58   63:           UTF8String
       :             'A-Trust Ges. f. Sicherheitssysteme im elektr. Da'
       :             'tenverkehr GmbH'
       :           }
       :         }
123   22:       SET {
125   20:         SEQUENCE {
127    3:           OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
132   13:           UTF8String 'a-sign-SSL-03'
       :           }
       :         }
147   22:       SET {
149   20:         SEQUENCE {
151    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
156   13:           UTF8String 'a-sign-SSL-03'
       :           }
       :         }
       :       }
171   30:     SEQUENCE {
173   13:       UTCTime 27/05/2013 11:46:01 GMT
188   13:       UTCTime 27/05/2018 09:46:01 GMT
       :       }
203   97:     SEQUENCE {
205   11:       SET {
207    9:         SEQUENCE {
209    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
214    2:           PrintableString 'AT'
       :           }
       :         }
218   16:       SET {
```

```
220   14:          SEQUENCE {
222    3:            OBJECT IDENTIFIER organizationName (2 5 4 10)
227    7:            UTF8String 'A-Trust'
         :            }
         :          }
236   16:        SET {
238   14:          SEQUENCE {
240    3:            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
245    7:            UTF8String 'A-Trust'
         :            }
         :          }
254   23:        SET {
256   21:          SEQUENCE {
258    3:            OBJECT IDENTIFIER commonName (2 5 4 3)
263   14:            UTF8String 'www.a-trust.at'
         :            }
         :          }
279   21:        SET {
281   19:          SEQUENCE {
283    3:            OBJECT IDENTIFIER serialNumber (2 5 4 5)
288   12:            PrintableString '694381007665'
         :            }
         :          }
         :        }
302  290:      SEQUENCE {
306   13:        SEQUENCE {
308    9:          OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
319    0:          NULL
         :          }
321  271:        BIT STRING, encapsulates {
326  266:          SEQUENCE {
330  257:            INTEGER
         :              00 BF 31 C2 8F 43 8B 97 FA E4 CE A4 8D DF 7B 39
         :              57 2C 6E 9D 3E 0D D2 2D E5 AB D5 F0 47 21 7C B9
         :              91 19 87 DB 1B 8A F6 D3 F9 7B D7 5C 29 36 0A 2C
         :              55 50 84 38 45 B7 A7 94 31 42 88 71 84 6A 58 8F
         :              83 4D 8F C2 E3 DE 87 EF 02 A5 94 D8 B2 EF 4F DE
         :              F9 61 E5 D6 FC 03 64 D5 68 35 7A E1 29 1E B3 44
         :              0A 5D 08 C8 A6 C6 34 22 3A CA F2 50 A2 EC 18 CB
         :              70 99 D5 D8 CC 70 BF F2 59 CD B4 00 C4 56 21 04
         :                      [ Another 129 bytes skipped ]
591    3:            INTEGER 65537
         :            }
         :          }
         :        }
596  572:      [3] {
600  568:        SEQUENCE {
604   17:          SEQUENCE {
606    3:            OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
611   10:            OCTET STRING, encapsulates {
613    8:              OCTET STRING 49 2C C2 54 A4 37 B4 0B
         :              }
         :            }
623   14:          SEQUENCE {
625    3:            OBJECT IDENTIFIER keyUsage (2 5 29 15)
630    1:            BOOLEAN TRUE
633    4:            OCTET STRING, encapsulates {
635    2:              BIT STRING 5 unused bits
         :                '101'B
         :              }
         :            }
639   19:          SEQUENCE {
641    3:            OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
646   12:            OCTET STRING, encapsulates {
648   10:              SEQUENCE {
```

```
116  650    8:                [0] 40 3E A1 D3 62 B4 03 DD
117           :                }
118           :              }
119           :            }
120  660  159:          SEQUENCE {
121  663    3:            OBJECT IDENTIFIER subjectAltName (2 5 29 17)
122  668  151:            OCTET STRING, encapsulates {
123  671  148:              SEQUENCE {
124  674   14:                [2] 'www.a-trust.at'
125  690   13:                [2] 'www.atrust.at'
126  705   10:                [2] 'a-trust.at'
127  717    9:                [2] 'atrust.at'
128  728   28:                [2] 'zda.sozialversicherung.gv.at'
129  758   36:                [2] 'zda-abnahme.sozialversicherung.gv.at'
130  796   24:                [1] 'servicecenter@a-trust.at'
131           :                }
132           :              }
133           :            }
134  822    9:          SEQUENCE {
135  824    3:            OBJECT IDENTIFIER basicConstraints (2 5 29 19)
136  829    2:            OCTET STRING, encapsulates {
137  831    0:              SEQUENCE {}
138           :              }
139           :            }
140  833  114:          SEQUENCE {
141  835    8:            OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
142  845  102:            OCTET STRING, encapsulates {
143  847  100:              SEQUENCE {
144  849   57:                SEQUENCE {
145  851    8:                  OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
146  861   45:                  [6]
147           :                    'http://www.a-trust.at/certs/a-sign-ssl-03.crt'
148           :                  }
149  908   39:                SEQUENCE {
150  910    8:                  OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
151  920   27:                  [6] 'http://ocsp.a-trust.at/ocsp'
152           :                  }
153           :                }
154           :              }
155           :            }
156  949   75:          SEQUENCE {
157  951    3:            OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
158  956   68:            OCTET STRING, encapsulates {
159  958   66:              SEQUENCE {
160  960   64:                SEQUENCE {
161  962    6:                  OBJECT IDENTIFIER '1 2 40 0 17 1 20'
162  970   54:                  SEQUENCE {
163  972   52:                    SEQUENCE {
164  974    8:                      OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
165  984   40:                      IA5String 'http://www.a-trust.at/docs/cp/a-sign-ssl'
166           :                      }
167           :                    }
168           :                  }
169           :                }
170           :              }
171           :            }
172 1026  143:          SEQUENCE {
173 1029    3:            OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
174 1034  135:            OCTET STRING, encapsulates {
175 1037  132:              SEQUENCE {
176 1040  129:                SEQUENCE {
177 1043  127:                  [0] {
178 1045  125:                    [0] {
179 1047  123:                      [6]
180           :                      'ldap://ldap.a-trust.at/ou=a-sign-SSL-03,o=A-Trus'
```

```
181          :                     't,c=AT?certificaterevocationlist?base?objectclas'
182          :                     's=eidCertificationAuthority'
183          :                       }
184          :                     }
185          :                   }
186          :                 }
187          :               }
188          :             }
189          :           }
190          :         }
191          :       }
192 1172   13:     SEQUENCE {
193 1174    9:       OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
194 1185    0:       NULL
195          :       }
196 1187  257:     BIT STRING
197          :       6F AC DA C7 43 3C F5 AC 72 25 1D 63 1C BD 00 FF
198          :       24 39 D3 1D 82 05 31 7C 3E 6C 67 70 39 CF B8 20
199          :       6A 6A AB C7 AF 3D 26 15 4B 55 F1 9A 00 2E 59 4F
200          :       73 04 4C BB 44 FB FF C1 70 6E 3C 8E B3 2A 74 3F
201          :       BB 0E C6 81 79 BF FD EC 66 38 65 29 3E E9 00 2A
202          :       52 4B 6C 9E C2 EB 00 DA F5 9E 84 79 40 47 47 81
203          :       44 90 D9 F1 FB E3 4D 0B EB 28 66 1E 5C 41 3C 35
204          :       E8 83 1C 31 D8 4A AB 73 14 E7 F6 64 F2 68 8F 6A
205          :               [ Another 128 bytes skipped ]
206          :     }
```

Listing A.1: Zertifikate mit mehreren Domainnamen im Subject Alternative Name

A.2 Zertifikate mit UPN im Subject Alternative Name

Der UPN befindet sich in der Zeile 110 und ist als unversalPrincipalName ausgewiesen.

```
1     0 1389: SEQUENCE {
2     4 1109:   SEQUENCE {
3     8    3:     [0] {
4    10    1:       INTEGER 2
5          :       }
6    13    3:     INTEGER 237364
7    18   13:     SEQUENCE {
8    20    9:       OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
9    31    0:       NULL
10         :       }
11   33  151:     SEQUENCE {
12   36   11:       SET {
13   38    9:         SEQUENCE {
14   40    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
15   45    2:           PrintableString 'AT'
16         :           }
17         :         }
18   49   72:       SET {
19   51   70:         SEQUENCE {
20   53    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
21   58   63:           UTF8String
22         :             'A-Trust Ges. f. Sicherheitssysteme im elektr. Da'
23         :             'tenverkehr GmbH'
24         :           }
25         :         }
26  123   30:       SET {
27  125   28:         SEQUENCE {
```

```
127    3:            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
132   21:            UTF8String 'a-sign-Premium-Enc-02'
         :            }
         :          }
155   30:        SET {
157   28:          SEQUENCE {
159    3:            OBJECT IDENTIFIER commonName (2 5 4 3)
164   21:            UTF8String 'a-sign-Premium-Enc-02'
         :            }
         :          }
         :        }
187   30:      SEQUENCE {
189   13:        UTCTime 05/02/2008 16:20:49 GMT
204   13:        UTCTime 05/02/2013 16:20:49 GMT
         :        }
219  118:      SEQUENCE {
221   11:        SET {
223    9:          SEQUENCE {
225    3:            OBJECT IDENTIFIER countryName (2 5 4 6)
230    2:            PrintableString 'AT'
         :            }
         :          }
234   27:        SET {
236   25:          SEQUENCE {
238    3:            OBJECT IDENTIFIER commonName (2 5 4 3)
243   18:            UTF8String 'Patrick Hagelkruys'
         :            }
         :          }
263   19:        SET {
265   17:          SEQUENCE {
267    3:            OBJECT IDENTIFIER surname (2 5 4 4)
272   10:            UTF8String 'Hagelkruys'
         :            }
         :          }
284   16:        SET {
286   14:          SEQUENCE {
288    3:            OBJECT IDENTIFIER givenName (2 5 4 42)
293    7:            UTF8String 'Patrick'
         :            }
         :          }
302   21:        SET {
304   19:          SEQUENCE {
306    3:            OBJECT IDENTIFIER serialNumber (2 5 4 5)
311   12:            PrintableString '369108117853'
         :            }
         :          }
325   12:        SET {
327   10:          SEQUENCE {
329    3:            OBJECT IDENTIFIER title (2 5 4 12)
334    3:            UTF8String 'Ing'
         :            }
         :          }
         :        }
339  223:      SEQUENCE {
342   13:        SEQUENCE {
344    9:          OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
355    0:          NULL
         :          }
357  205:        BIT STRING, encapsulates {
361  201:          SEQUENCE {
364  193:            INTEGER
         :              00 DA 25 BE FE 05 DE 74 AC A8 F3 EE E4 EB 7E B7
         :              69 07 CD 6C C1 30 C8 94 08 A1 5B EC E5 A5 B2 3A
         :              45 B5 04 6C CA 09 1C 07 47 21 93 0C 38 AF 6F D2
         :              3D E6 C7 CA A1 B9 0E 9B CE 76 50 CC 27 C3 80 BB
```

```
93          :              69 1E 1D A9 EA 17 9A 1A FF 7F 50 C7 2A 80 E5 C0
94          :              7D 2D 94 B9 10 1E 08 B8 1A 6A 53 21 47 B4 FD F6
95          :              A2 8D DC 61 3B A8 CF 3F BC 05 83 83 FE 06 53 18
96          :              82 69 E4 8C C5 38 61 6A 52 67 66 BE 6A 63 46 9A
97          :                      [ Another 65 bytes skipped ]
98  560     3:            INTEGER 65537
99          :            }
100         :          }
101         :        }
102 565   548:      [3] {
103 569   544:        SEQUENCE {
104 573    71:          SEQUENCE {
105 575     3:            OBJECT IDENTIFIER subjectAltName (2 5 29 17)
106 580    64:            OCTET STRING, encapsulates {
107 582    62:              SEQUENCE {
108 584    37:                [0] {
109 586    10:                  OBJECT IDENTIFIER
110         :                    universalPrincipalName (1 3 6 1 4 1 311 20 2 3)
111 598    23:                  [0] {
112 600    21:                    UTF8String 'hagelkruys@a-trust.at'
113         :                    }
114         :                  }
115 623    21:                [1] 'hagelkruys@a-trust.at'
116         :                }
117         :              }
118         :            }
119 646    41:          SEQUENCE {
120 648     3:            OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
121 653    34:            OCTET STRING, encapsulates {
122 655    32:              SEQUENCE {
123 657     8:                OBJECT IDENTIFIER clientAuth (1 3 6 1 5 5 7 3 2)
124 667    10:                OBJECT IDENTIFIER
125         :                  smartcardLogon (1 3 6 1 4 1 311 20 2 2)
126 679     8:                OBJECT IDENTIFIER emailProtection (1 3 6 1 5 5 7 3 4)
127         :                }
128         :              }
129         :            }
130 689    19:          SEQUENCE {
131 691     3:            OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
132 696    12:            OCTET STRING, encapsulates {
133 698    10:              SEQUENCE {
134 700     8:                [0] 47 21 47 8E 97 61 E3 1E
135         :                }
136         :              }
137         :            }
138 710   123:          SEQUENCE {
139 712     8:            OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
140 722   111:            OCTET STRING, encapsulates {
141 724   109:              SEQUENCE {
142 726    66:                SEQUENCE {
143 728     8:                  OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
144 738    54:                  [6]
145         :                    'http://www.a-trust.at/certs/a-sign-Premium-Enc-0'
146         :                    '2a.crt'
147         :                  }
148 794    39:                SEQUENCE {
149 796     8:                  OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
150 806    27:                  [6] 'http://ocsp.a-trust.at/ocsp'
151         :                  }
152         :                }
153         :              }
154         :            }
155 835    77:          SEQUENCE {
156 837     3:            OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
157 842    70:            OCTET STRING, encapsulates {
```

```
158  844   68:              SEQUENCE {
159  846   66:                SEQUENCE {
160  848    6:                  OBJECT IDENTIFIER '1 2 40 0 17 1 12'
161  856   56:                  SEQUENCE {
162  858   54:                    SEQUENCE {
163  860    8:                      OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
164  870   42:                      IA5String
165       :                      'http://www.a-trust.at/docs/cp/a-sign-token'
166       :                        }
167       :                      }
168       :                    }
169       :                  }
170       :                }
171       :              }
172  914  154:            SEQUENCE {
173  917    3:              OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
174  922  146:              OCTET STRING, encapsulates {
175  925  143:                SEQUENCE {
176  928  140:                  SEQUENCE {
177  931  137:                    [0] {
178  934  134:                      [0] {
179  937  131:                        [6]
180       :                      'ldap://ldap.a-trust.at/ou=a-sign-Premium-Enc-02,'
181       :                      'o=A-Trust,c=AT?certificaterevocationlist?base?ob'
182       :                      'jectclass=eidCertificationAuthority'
183       :                        }
184       :                      }
185       :                    }
186       :                  }
187       :                }
188       :              }
189 1071   17:            SEQUENCE {
190 1073    3:              OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
191 1078   10:              OCTET STRING, encapsulates {
192 1080    8:                OCTET STRING 4A 31 52 E5 C8 9E E7 7B
193       :                }
194       :              }
195 1090   14:            SEQUENCE {
196 1092    3:              OBJECT IDENTIFIER keyUsage (2 5 29 15)
197 1097    1:              BOOLEAN TRUE
198 1100    4:              OCTET STRING, encapsulates {
199 1102    2:                BIT STRING 4 unused bits
200       :                  '1101'B
201       :                }
202       :              }
203 1106    9:            SEQUENCE {
204 1108    3:              OBJECT IDENTIFIER basicConstraints (2 5 29 19)
205 1113    2:              OCTET STRING, encapsulates {
206 1115    0:                SEQUENCE {}
207       :                }
208       :              }
209       :            }
210       :          }
211       :        }
212 1117   13:    SEQUENCE {
213 1119    9:      OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
214 1130    0:      NULL
215       :      }
216 1132  257:    BIT STRING
217       :      20 7C EF DE FF 9A B9 39 1D 96 AF 3F 51 26 D7 D2
218       :      02 DF D0 93 91 E1 77 46 D9 27 B3 EB 24 05 FC 47
219       :      F1 FB FA 23 ED 34 2C 6F CB 6D 4A C5 58 84 7A AA
220       :      AC 8C EF 49 AC 6E B7 9F 1B B0 27 67 65 E2 D0 A8
221       :      2D B2 75 F2 23 23 B8 3C 6C AF 11 19 BA 95 88 FA
222       :      D4 CE A5 9C C9 E6 51 C5 8E 53 6C 03 88 E3 C3 FB
```

```
223          :      D8 71 A5 48 15 5D 1A A1 87 43 73 58 0D 63 88 4F
224          :      CF ED 31 7A C5 6A 8E F8 15 40 F1 2E 0B 17 F0 98
225          :               [ Another 128 bytes skipped ]
226          :    }
```

Listing A.2: Zertifikate mit UPN im Subject Alternative Name

A.3 Zertifikate mit Qualified Certificate Statement

Das Qualified Certificate Statement befindet sich in der Zeile 98 und ist als qcStatement ausgewiesen.

```
1      0 1211: SEQUENCE {
2      4  931:   SEQUENCE {
3      8    3:     [0] {
4     10    1:       INTEGER 2
5            :       }
6     13    3:     INTEGER 418499
7     18   13:     SEQUENCE {
8     20    9:       OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
9     31    0:       NULL
10           :       }
11    33  151:     SEQUENCE {
12    36   11:       SET {
13    38    9:         SEQUENCE {
14    40    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
15    45    2:           PrintableString 'AT'
16           :           }
17           :         }
18    49   72:       SET {
19    51   70:         SEQUENCE {
20    53    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
21    58   63:           UTF8String
22           :             'A-Trust Ges. f. Sicherheitssysteme im elektr. Da'
23           :             'tenverkehr GmbH'
24           :           }
25           :         }
26   123   30:       SET {
27   125   28:         SEQUENCE {
28   127    3:           OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
29   132   21:           UTF8String 'a-sign-Premium-Sig-02'
30           :           }
31           :         }
32   155   30:       SET {
33   157   28:         SEQUENCE {
34   159    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
35   164   21:           UTF8String 'a-sign-Premium-Sig-02'
36           :           }
37           :         }
38           :       }
39   187   30:     SEQUENCE {
40   189   13:       UTCTime 21/01/2010 17:08:09 GMT
41   204   13:       UTCTime 21/01/2015 17:08:09 GMT
42           :       }
43   219  104:     SEQUENCE {
44   221   11:       SET {
45   223    9:         SEQUENCE {
46   225    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
47   230    2:           PrintableString 'AT'
48           :           }
49           :         }
```

```
234   27:        SET {
236   25:          SEQUENCE {
238    3:            OBJECT IDENTIFIER commonName (2 5 4 3)
243   18:            UTF8String 'Patrick Hagelkruys'
        :            }
        :          }
263   19:        SET {
265   17:          SEQUENCE {
267    3:            OBJECT IDENTIFIER surname (2 5 4 4)
272   10:            UTF8String 'Hagelkruys'
        :            }
        :          }
284   16:        SET {
286   14:          SEQUENCE {
288    3:            OBJECT IDENTIFIER givenName (2 5 4 42)
293    7:            UTF8String 'Patrick'
        :            }
        :          }
302   21:        SET {
304   19:          SEQUENCE {
306    3:            OBJECT IDENTIFIER serialNumber (2 5 4 5)
311   12:            PrintableString '374315638980'
        :            }
        :          }
        :        }
325   89:      SEQUENCE {
327   19:        SEQUENCE {
329    7:          OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
338    8:          OBJECT IDENTIFIER prime256v1 (1 2 840 10045 3 1 7)
        :          }
348   66:        BIT STRING
        :          04 11 DE 52 95 DA A3 90 FC FD B3 81 CC C4 B1 63
        :          0A 97 EC D2 DA A9 38 CA 43 5D E8 4E F5 60 C3 CC
        :          04 CD D9 07 AD 1F 0A EC 25 84 63 74 86 7E 64 26
        :          8E 09 C1 EB EB 5F A3 32 85 06 E1 71 A3 85 E2 8C
        :          61
        :        }
416  519:      [3] {
420  515:        SEQUENCE {
424   19:          SEQUENCE {
426    3:            OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
431   12:            OCTET STRING, encapsulates {
433   10:              SEQUENCE {
435    8:                [0] 4D DF E1 FF 4B D9 C9 DF
        :                }
        :              }
        :            }
445   39:          SEQUENCE {
447    8:            OBJECT IDENTIFIER qcStatements (1 3 6 1 5 5 7 1 3)
457    1:            BOOLEAN TRUE
460   24:            OCTET STRING, encapsulates {
462   22:              SEQUENCE {
464    8:                SEQUENCE {
466    6:                  OBJECT IDENTIFIER etsiQcsCompliance (0 4 0 1862 1 1)
        :                  }
474   10:                SEQUENCE {
476    8:                  OBJECT IDENTIFIER
        :                    pkixQCSyntax-v1 (1 3 6 1 5 5 7 11 1)
        :                  }
        :                }
        :              }
        :            }
486  123:          SEQUENCE {
488    8:            OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
498  111:            OCTET STRING, encapsulates {
```

```
115  500  109:              SEQUENCE {
116  502   66:                SEQUENCE {
117  504    8:                  OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
118  514   54:                  [6]
119       :                    'http://www.a-trust.at/certs/a-sign-Premium-Sig-0'
120       :                    '2a.crt'
121       :                  }
122  570   39:                SEQUENCE {
123  572    8:                  OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
124  582   27:                  [6] 'http://ocsp.a-trust.at/ocsp'
125       :                  }
126       :                }
127       :              }
128       :            }
129  611   89:          SEQUENCE {
130  613    3:            OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
131  618   82:            OCTET STRING, encapsulates {
132  620   80:              SEQUENCE {
133  622   68:                SEQUENCE {
134  624    6:                  OBJECT IDENTIFIER '1 2 40 0 17 1 11'
135  632   58:                  SEQUENCE {
136  634   56:                    SEQUENCE {
137  636    8:                      OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
138  646   44:                      IA5String
139       :                      'http://www.a-trust.at/docs/cp/a-sign-Premium'
140       :                      }
141       :                    }
142       :                  }
143  692    8:                SEQUENCE {
144  694    6:                  OBJECT IDENTIFIER '0 4 0 1456 1 1'
145       :                  }
146       :                }
147       :              }
148       :            }
149  702  154:          SEQUENCE {
150  705    3:            OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
151  710  146:            OCTET STRING, encapsulates {
152  713  143:              SEQUENCE {
153  716  140:                SEQUENCE {
154  719  137:                  [0] {
155  722  134:                    [0] {
156  725  131:                      [6]
157       :                      'ldap://ldap.a-trust.at/ou=a-sign-Premium-Sig-02,'
158       :                      'o=A-Trust,c=AT?certificaterevocationlist?base?ob'
159       :                      'jectclass=eidCertificationAuthority'
160       :                      }
161       :                    }
162       :                  }
163       :                }
164       :              }
165       :            }
166  859   17:          SEQUENCE {
167  861    3:            OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
168  866   10:            OCTET STRING, encapsulates {
169  868    8:              OCTET STRING 4B 85 BE B9 23 CD 2E 39
170       :              }
171       :            }
172  878   14:          SEQUENCE {
173  880    3:            OBJECT IDENTIFIER keyUsage (2 5 29 15)
174  885    1:            BOOLEAN TRUE
175  888    4:            OCTET STRING, encapsulates {
176  890    2:              BIT STRING 6 unused bits
177       :                '11'B
178       :              }
179       :            }
```

```
180  894   32:          SEQUENCE {
181  896    3:            OBJECT IDENTIFIER subjectAltName (2 5 29 17)
182  901   25:            OCTET STRING, encapsulates {
183  903   23:              SEQUENCE {
184  905   21:                [1] 'hagelkruys@a-trust.at'
185        :                }
186        :              }
187        :            }
188  928    9:          SEQUENCE {
189  930    3:            OBJECT IDENTIFIER basicConstraints (2 5 29 19)
190  935    2:            OCTET STRING, encapsulates {
191  937    0:              SEQUENCE {}
192        :              }
193        :            }
194        :          }
195        :        }
196        :      }
197  939   13:    SEQUENCE {
198  941    9:      OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
199  952    0:      NULL
200        :      }
201  954  257:    BIT STRING
202        :      9D A0 93 FA 41 14 9D 15 8D D3 CB 9B EE E6 AD C1
203        :      E8 20 BB F0 63 4F 32 B5 55 60 85 BE 75 2D 44 71
204        :      EC D4 68 EF 3E 47 87 6B 54 E9 88 F8 C6 C3 32 98
205        :      D7 67 88 F1 4F 07 E8 5B 51 83 CA 32 A8 23 CF A6
206        :      9C A2 8D 38 20 33 44 8A DF A3 04 A2 28 39 D8 80
207        :      EF 6F E6 27 D4 13 69 7E E3 BA 5E C3 3F E6 9A 05
208        :      DE 07 1C 80 1F D3 11 8D 1A 48 1B C1 D7 03 4A FC
209        :      21 82 4B 79 F6 78 37 8C 3F CC B0 8B A0 AE 85 5B
210        :              [ Another 128 bytes skipped ]
211        :    }
```

Listing A.3: Zertifikate mit Qualified Certificate Statement

A.4 Zertifikate mit Dienstleistereigenschaft

Die Dienstleistereigenschaft befindet sich in der Zeile 127. Da es sich hierbei um eine spezielle Erweiterung für den österreichischen Raum handelt, wurde kein Name erkannt und stattdessen nur der Object Identifier 1.2.40.0.10.1.1.2 angegeben.

```
1     0 1298: SEQUENCE {
2     4 1018:   SEQUENCE {
3     8    3:     [0] {
4    10    1:       INTEGER 2
5          :       }
6    13    3:     INTEGER 654183
7    18   13:     SEQUENCE {
8    20    9:       OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
9    31    0:       NULL
10         :       }
11   33  135:     SEQUENCE {
12   36   11:       SET {
13   38    9:         SEQUENCE {
14   40    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
15   45    2:           PrintableString 'AT'
16         :           }
17         :         }
18   49   72:       SET {
19   51   70:         SEQUENCE {
```

```
 53    3:            OBJECT IDENTIFIER organizationName (2 5 4 10)
 58   63:            UTF8String
        :              'A-Trust Ges. f. Sicherheitssysteme im elektr. Da'
        :              'tenverkehr GmbH'
        :            }
        :          }
123   22:        SET {
125   20:          SEQUENCE {
127    3:            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
132   13:            UTF8String 'a-sign-SSL-03'
        :            }
        :          }
147   22:        SET {
149   20:          SEQUENCE {
151    3:            OBJECT IDENTIFIER commonName (2 5 4 3)
156   13:            UTF8String 'a-sign-SSL-03'
        :            }
        :          }
        :        }
171   30:      SEQUENCE {
173   13:        UTCTime 04/11/2011 08:27:59 GMT
188   13:        UTCTime 04/11/2016 07:27:59 GMT
        :        }
203   98:      SEQUENCE {
205   11:        SET {
207    9:          SEQUENCE {
209    3:            OBJECT IDENTIFIER countryName (2 5 4 6)
214    2:            PrintableString 'AT'
        :            }
        :          }
218   16:        SET {
220   14:          SEQUENCE {
222    3:            OBJECT IDENTIFIER organizationName (2 5 4 10)
227    7:            UTF8String 'A-Trust'
        :            }
        :          }
236   16:        SET {
238   14:          SEQUENCE {
240    3:            OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
245    7:            UTF8String 'A-Trust'
        :            }
        :          }
254   24:        SET {
256   22:          SEQUENCE {
258    3:            OBJECT IDENTIFIER commonName (2 5 4 3)
263   15:            UTF8String 'moa3.a-trust.at'
        :            }
        :          }
280   21:        SET {
282   19:          SEQUENCE {
284    3:            OBJECT IDENTIFIER serialNumber (2 5 4 5)
289   12:            PrintableString '893875030700'
        :            }
        :          }
        :        }
303  290:      SEQUENCE {
307   13:        SEQUENCE {
309    9:          OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
320    0:          NULL
        :          }
322  271:        BIT STRING, encapsulates {
327  266:          SEQUENCE {
331  257:            INTEGER
        :              00 D8 BC 43 C9 13 9E B8 0B F7 25 64 5B 6C D8 B8
        :              43 DE B9 C6 EF 15 F3 27 5A DE 0D F9 AD 2D 58 75
```

```
85          :              5F 32 97 F2 AD 44 3A 58 49 59 C8 0A 02 05 9D EE
86          :              11 33 F9 E3 71 AE 23 ED 12 6A A9 73 11 BA 36 AE
87          :              30 F1 FB BA 5B 6C 4E 92 DE 28 32 1E 8B 89 05 A6
88          :              74 FB DB CB 9A BE 79 DC B5 49 AA 85 3F 87 CE F8
89          :              31 EA EA 48 B8 BA ED EA 81 46 98 DC D1 62 B1 07
90          :              80 21 FF FC 54 40 F6 CF 1D 08 36 56 39 10 85 1F
91          :                      [ Another 129 bytes skipped ]
92  592     3:            INTEGER 65537
93          :            }
94          :          }
95          :        }
96  597   425:      [3] {
97  601   421:        SEQUENCE {
98  605    17:          SEQUENCE {
99  607     3:            OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
100 612    10:            OCTET STRING, encapsulates {
101 614     8:              OCTET STRING 47 87 69 44 EC 54 AC 4B
102         :              }
103         :            }
104 624    14:          SEQUENCE {
105 626     3:            OBJECT IDENTIFIER keyUsage (2 5 29 15)
106 631     1:            BOOLEAN TRUE
107 634     4:            OCTET STRING, encapsulates {
108 636     2:              BIT STRING 5 unused bits
109         :                '101'B
110         :              }
111         :            }
112 640    19:          SEQUENCE {
113 642     3:            OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
114 647    12:            OCTET STRING, encapsulates {
115 649    10:              SEQUENCE {
116 651     8:                [0] 40 3E A1 D3 62 B4 03 DD
117         :                }
118         :              }
119         :            }
120 661     9:          SEQUENCE {
121 663     3:            OBJECT IDENTIFIER basicConstraints (2 5 29 19)
122 668     2:            OCTET STRING, encapsulates {
123 670     0:              SEQUENCE {}
124         :              }
125         :            }
126 672    13:          SEQUENCE {
127 674     7:            OBJECT IDENTIFIER '1 2 40 0 10 1 1 2'
128 683     2:            OCTET STRING, encapsulates {
129 685     0:              NULL
130         :              }
131         :            }
132 687   114:          SEQUENCE {
133 689     8:            OBJECT IDENTIFIER authorityInfoAccess (1 3 6 1 5 5 7 1 1)
134 699   102:            OCTET STRING, encapsulates {
135 701   100:              SEQUENCE {
136 703    57:                SEQUENCE {
137 705     8:                  OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
138 715    45:                  [6]
139         :                    'http://www.a-trust.at/certs/a-sign-ssl-03.crt'
140         :                  }
141 762    39:                SEQUENCE {
142 764     8:                  OBJECT IDENTIFIER ocsp (1 3 6 1 5 5 7 48 1)
143 774    27:                  [6] 'http://ocsp.a-trust.at/ocsp'
144         :                  }
145         :                }
146         :              }
147         :            }
148 803    75:          SEQUENCE {
149 805     3:            OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
```

```
150  810   68:            OCTET STRING, encapsulates {
151  812   66:              SEQUENCE {
152  814   64:                SEQUENCE {
153  816    6:                  OBJECT IDENTIFIER '1 2 40 0 17 1 20'
154  824   54:                  SEQUENCE {
155  826   52:                    SEQUENCE {
156  828    8:                      OBJECT IDENTIFIER cps (1 3 6 1 5 5 7 2 1)
157  838   40:                      IA5String 'http://www.a-trust.at/docs/cp/a-sign-ssl'
158       :                      }
159       :                    }
160       :                  }
161       :                }
162       :              }
163       :            }
164  880  143:          SEQUENCE {
165  883    3:            OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
166  888  135:            OCTET STRING, encapsulates {
167  891  132:              SEQUENCE {
168  894  129:                SEQUENCE {
169  897  127:                  [0] {
170  899  125:                    [0] {
171  901  123:                      [6]
172       :                      'ldap://ldap.a-trust.at/ou=a-sign-SSL-03,o=A-Trus'
173       :                      't,c=AT?certificaterevocationlist?base?objectclas'
174       :                      's=eidCertificationAuthority'
175       :                      }
176       :                    }
177       :                  }
178       :                }
179       :              }
180       :            }
181       :          }
182       :        }
183       :      }
184 1026   13:    SEQUENCE {
185 1028    9:      OBJECT IDENTIFIER sha1WithRSAEncryption (1 2 840 113549 1 1 5)
186 1039    0:      NULL
187       :      }
188 1041  257:    BIT STRING
189       :      0E 5E 07 B7 A8 45 7A 1E E3 2D 2E B5 70 D9 84 E1
190       :      C2 23 48 2A 16 9A 1A 8F 98 A3 75 C4 CC F8 9F C3
191       :      30 27 F1 41 06 94 52 CD FB 90 4B 93 97 C9 77 4A
192       :      33 CB 36 07 24 2A 01 A1 88 0F 49 DF B6 CB 31 A9
193       :      4A 69 CA 30 27 69 80 15 E2 D5 6E D9 AD 99 C3 F3
194       :      2C F8 19 CA D6 EC 3C BC BD 38 2C 0B DD D8 B2 67
195       :      1E CA 3B 29 41 A8 56 1B 1B D8 83 76 E7 7B 90 EB
196       :      57 1B 4D 7B 36 23 AB C4 81 9F BB 9E A1 18 C2 92
197       :              [ Another 128 bytes skipped ]
198       :    }
```

Listing A.4: Zertifikate mit Dienstleistereigenschaft

B EJBCA

Java-Properties-Eintrag für Dienstleistereigenschaft

```
id1.oid = 1.2.40.0.10.1.1.2
id1.classpath=org.cesecore.certificates.certificate.certextensions.
    BasicCertificateExtension
id1.displayname=DienstleisterExtension
id1.used=true
id1.translatable=false
id1.critical=false
id1.property.dynamic=false
id1.property.encoding=DERNULL
id1.property.value=ignored
```

Listing B.1: Java-Properties-Eintrag für Dienstleistereigenschaft

QC-Statement mit pkixQCSyntax-v2

```
SEQUENCE {
  OBJECT IDENTIFIER qcStatements (1 3 6 1 5 5 7 1 3)
  BOOLEAN TRUE
  OCTET STRING, encapsulates {
    SEQUENCE {
      SEQUENCE {
        OBJECT IDENTIFIER '1 3 6 1 5 5 7 11 2'
        }
      SEQUENCE {
        OBJECT IDENTIFIER etsiQcsCompliance (0 4 0 1862 1 1)
        }
      }
    }
  }
```

Listing B.2: QC-Statement mit pkixQCSyntax-v2

C XCA-Einstellungen für Testzertifikate

Zertifikate mit mehreren Domainnamen im Subject Alternative Name

```
authorityInfoAccess=@aia_sect
certificatePolicies=ia5org,@polsect
crlDistributionPoints=@crl_section

[polsect]
policyIdentifier=1.2.40.0.17.1.20
CPS.1="http://www.a-trust.at/docs/cp/a-sign-ssl"

[aia_sect]
caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-ssl-03.crt
OCSP;URI.1=http://ocsp.a-trust.at/OCSP

[crl_section]
URI.1=ldap://ldap.a-trust.at/out=a-sign-SSL-03,o=A-Trust,c=AT?
    certificaterevocationlist?base?objectclass=eidCertificateAuthority
```

Listing C.1: XCA-Zertifikate mit mehreren Domainnamen im Subject Alternative Name

Zertifikate mit UPN im Subject Alternative Name

```
authorityInfoAccess=@aia_sect
certificatePolicies=ia5org,@polsect
crlDistributionPoints=@crl_section

[polsect]
policyIdentifier=1.2.40.0.17.1.12
CPS.1="http://www.a-trust.at/docs/cp/a-sign-token"

[aia_sect]
caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-Premium-Sig-02a.crt
OCSP;URI.1=http://ocsp.a-trust.at/ocsp

[crl_section]
URI.1=ldap://ldap.a-trust.at/ou=a-sign-Premium-Enc-02,o=A-Trust,c=AT?
    certificaterevocationlist?base?objectclass=eidCertificationAuthority
```

Listing C.2: XCA-Zertifikate mit UPN im Subject Alternative Name

Für den UPN muss in der Benutzeroberfläche im Feld Subject Alternative Name – other-Name folgender Wert eingetragen werden:

```
1.3.6.1.4.1.311.20.2.3;UTF8:hagelkruys@a-trust.at
```

Listing C.3: XCA-Zertifikate mit UPN im Subject Alternative Name - UPN

Zertifikate mit Qualified Certificate Statement

```
authorityInfoAccess=@aia_sect
certificatePolicies=ia5org,@polsect,0.4.0.1456.1.1
crlDistributionPoints=@crl_section

1.3.6.1.5.5.7.1.3=critical,ASN1:SEQUENCE:seq_sect

[seq_sect]
field1=SEQUENCE:seq_sect1
field2=SEQUENCE:seq_sect2

[seq_sect1]
field1=OID:0.4.0.1862.1.1

[seq_sect2]
field1=OID:1.3.6.1.5.5.7.11.1

[polsect]
policyIdentifier=1.2.40.0.17.1.11
CPS.1="http://www.a-trust.at/docs/cp/a-sign-Premium"

[aia_sect]
caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-Premium-Sig-02a.crt
OCSP;URI.1=http://ocsp.a-trust.at/ocsp

[crl_section]
URI.1=ldap://ldap.a-trust.at/ou=a-sign-Premium-Sig-02,o=A-Trust,c=AT?
    certificaterevocationlist?base?objectclass=eidCertificationAuthority
```

Listing C.4: XCA-Zertifikate mit Qualified Certificate Statement

Zertifikate mit Dienstleistereigenschaft

```
authorityInfoAccess=@aia_sect
certificatePolicies=ia5org,@polsect
crlDistributionPoints=@crl_section

1.2.40.0.10.1.1.2=ASN1:NULL

[polsect]
policyIdentifier=1.2.40.0.17.1.20
CPS.1="http://www.a-trust.at/docs/cp/a-sign-ssl"

[aia_sect]
caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-ssl-03.crt
OCSP;URI.1=http://ocsp.a-trust.at/OCSP

[crl_section]
URI.1=ldap://ldap.a-trust.at/out=a-sign-SSL-03,o=A-Trust,c=AT?
    certificaterevocationlist?base?objectclass=eidCertificateAuthority
```

Listing C.5: XCA-Zertifikate mit Dienstleistereigenschaft

D OpenSSL

Erzeugen des CA-Root-Zertifikats

Privaten Schlüssel für das CA-Zertifikat erzeugen

```
1 OpenSSL\bin\openssl.exe genrsa -des3 -out ca\private\root-ca.key 2048
```

Listing D.1: Privaten CA-Schlüssel erzeugen

Zertifikat zum zuvor erstellen privaten Schlüssel erzeugen

```
1 OpenSSL\bin\openssl.exe req -new -x509 -days 3650 -key ca\private\root-ca.key -out
      ca\certs\root-ca.crt -config cfg\openssl.cfg
```

Listing D.2: CA-Zertifikat erzeugen

Erzeugen des Benutzerzertifikats

Erzeugen eines privaten Schlüssels und eines Zertifikat-Requests. Der Request wird anschließend von dem privaten Schlüssel der CA signiert. Für jedes der Testzertifikate wird ein eigener privater Schlüssel und ein Zertifikats-Request erstellt.

```
1 OpenSSL\bin\openssl.exe req -newkey rsa:2048 -keyout ca\userkeys\test.key -config
      cfg\openssl.cfg -out ca\userreq\test-request.req
```

Listing D.3: Erzeugen des privaten Schlüssels und eines Zertifikat-Requests des Benutzers

Erzeugen des Zertifikats des Benutzers, wobei für jedes der Zertifikate eine eigene OpenSSL-Extension-Datei verwendet wurde. Die Argumente `-extfile` und `-extensions` sind für die jeweiligen Zertifikate anzupassen.

```
1 OpenSSL\bin\openssl.exe  ca -name CA_default -verbose -config cfg\openssl.cfg -in
      ca\userreq\test-request.req -extfile cfg\extensions.cfg -extensions upn_ext -
      out test.cer
```

Listing D.4: Erzeugen des Zertifikats des Benutzers

Extension-Datei: Zertifikat mit mehreren Domainnamen im Subject Alternative Name

```
1
2 openssl_conf = openssl_init
3
4 [openssl_init]
5 oid_section = new_oids
```

```
6  oid_file= C:/OpenSSL-CA/oid_file.TXT
7
8  [new_oids]
9  serialNumber = 2.5.4.5
10
11 [multi_domain_ext]
12 basicConstraints = CA:FALSE
13 keyUsage=critical, digitalSignature, keyEncipherment
14 subjectKeyIdentifier=hash
15 authorityKeyIdentifier=keyid,issuer
16 subjectAltName=@altname
17 authorityInfoAccess=@aia_sect
18 certificatePolicies=ia5org,@polsect
19 crlDistributionPoints=@crl_section
20
21 [polsect]
22 policyIdentifier=1.2.40.0.17.1.20
23 CPS.1="http://www.a-trust.at/docs/cp/a-sign-ssl"
24
25 [aia_sect]
26 caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-ssl-03.crt
27 OCSP;URI.1=http://ocsp.a-trust.at/OCSP
28
29 [crl_section]
30 URI.1=ldap://ldap.a-trust.at/out=a-sign-SSL-03,o=A-Trust,c=AT?
       certificaterevocationlist?base?objectclass=eidCertificateAuthority
31
32 [altname]
33 DNS.1=www.a-trust.at
34 DNS.2=www.atrust.at
35 DNS.3=a-trust.at
36 DNS.4=atrust.at
37 DNS.5=zda.sozialversicherung.gv.at
38 DNS.6=zda-abnahme.sozialversicherung.gv.at
39 email=servicecenter@a-trust.at
```

Listing D.5: OpenSSL-Extension-Datei website_subject_alternative_name_multiple_urls.cfg

Extension-Datei: Zertifikat mit UPN im Subject Alternative Name

```
1
2  openssl_conf = openssl_init
3
4  [openssl_init]
5  oid_section = new_oids
6  oid_file= C:/OpenSSL-CA/oid_file.TXT
7
8
9  [new_oids]
10 serialNumber = 2.5.4.5
11
12 [upn_ext]
13 basicConstraints = CA:FALSE
14 keyUsage=critical, digitalSignature, keyEncipherment, dataEncipherment
15 subjectKeyIdentifier=hash
16 authorityKeyIdentifier=keyid,issuer
17 authorityInfoAccess=@aia_sect
18 certificatePolicies=ia5org,@polsect
19 crlDistributionPoints=@crl_section
20 subjectAltName=@altname
```

```
extendedKeyUsage = clientAuth,emailProtection,1.3.6.1.4.1.311.20.2.2

[polsect]
policyIdentifier=1.2.40.0.17.1.12
CPS.1="http://www.a-trust.at/docs/cp/a-sign-token"

[aia_sect]
caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-Premium-Sig-02a.crt
OCSP;URI.1=http://ocsp.a-trust.at/ocsp

[crl_section]
URI.1=ldap://ldap.a-trust.at/ou=a-sign-Premium-Enc-02,o=A-Trust,c=AT?
    certificaterevocationlist?base?objectclass=eidCertificationAuthority


[altname]
otherName=1.3.6.1.4.1.311.20.2.3;UTF8:hagelkruys@a-trust.at
email=hagelkruys@a-trust.at
```

Listing D.6: OpenSSL-Extension-Datei subject_alternative_name_upn.cfg

Extension-Datei: Zertifikate mit Qualified Certificate Statement

```
openssl_conf = openssl_init

[openssl_init]
oid_section = new_oids
oid_file= C:/OpenSSL-CA/oid_file.TXT


[new_oids]
serialNumber = 2.5.4.5

[qc_ext]
basicConstraints = CA:FALSE
keyUsage=critical, digitalSignature, nonRepudiation
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
authorityInfoAccess=@aia_sect
certificatePolicies=ia5org,@polsect,0.4.0.1456.1.1
crlDistributionPoints=@crl_section
1.3.6.1.5.5.7.1.3=critical,ASN1:SEQUENCE:seq_sect
subjectAltName=@altname

[seq_sect]
field1=SEQUENCE:seq_sect1
field2=SEQUENCE:seq_sect2

[seq_sect1]
field1=OID:0.4.0.1862.1.1

[seq_sect2]
field1=OID:1.3.6.1.5.5.7.11.1

[polsect]
policyIdentifier=1.2.40.0.17.1.11
CPS.1="http://www.a-trust.at/docs/cp/a-sign-Premium"

[aia_sect]
caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-Premium-Sig-02a.crt
```

```
38 OCSP;URI.1=http://ocsp.a-trust.at/ocsp
39
40 [crl_section]
41 URI.1=ldap://ldap.a-trust.at/ou=a-sign-Premium-Sig-02,o=A-Trust,c=AT?
       certificaterevocationlist?base?objectclass=eidCertificationAuthority
42
43
44 [altname]
45 email=hagelkruys@a-trust.at
```

Listing D.7: OpenSSL-Extension-Datei qualified_certificate_statement.cfg

Extension-Datei: Zertifikate mit Dienstleistereigenschaft

```
1
2  openssl_conf = openssl_init
3
4  [openssl_init]
5  oid_section = new_oids
6  oid_file= C:/OpenSSL-CA/oid_file.TXT
7
8  [new_oids]
9  serialNumber = 2.5.4.5
10
11 [dienstleister_ext]
12 basicConstraints = CA:FALSE
13 keyUsage=critical, digitalSignature, keyEncipherment
14 subjectKeyIdentifier=hash
15 authorityKeyIdentifier=keyid,issuer
16 authorityInfoAccess=@aia_sect
17 certificatePolicies=ia5org,@polsect
18 crlDistributionPoints=@crl_section
19 1.2.40.0.10.1.1.2=ASN1:NULL
20
21 [polsect]
22 policyIdentifier=1.2.40.0.17.1.20
23 CPS.1="http://www.a-trust.at/docs/cp/a-sign-ssl"
24
25 [aia_sect]
26 caIssuers;URI.1=http://www.a-trust.at/certs/a-sign-ssl-03.crt
27 OCSP;URI.1=http://ocsp.a-trust.at/OCSP
28
29 [crl_section]
30 URI.1=ldap://ldap.a-trust.at/out=a-sign-SSL-03,o=A-Trust,c=AT?
       certificaterevocationlist?base?objectclass=eidCertificateAuthority
```

Listing D.8: OpenSSL-Extension-Datei webserver_dienstleistereigenschaft.cfg

E r509 Ruby Certificate Authority

Allgemeiner Aufbau der Konfigurationsdatei

```
---
certificate_authorities:
  r509_ca:
    ca_cert:
      cert: root.cer
      key: root.key
    ocsp_start_skew_seconds: 3600
    ocsp_validity_hours: 168
    crl_list_file: r509_ca_list.txt
    crl_number_file: r509_ca_crlnumber.txt
    crl_validity_hours: 168
    crl_md: SHA1
    custom_oids:
    - :oid: 1.2.40.0.10.1.1.2
      :short_name: dienstleister
    - :oid: 1.3.6.1.4.1.311.20.2.2
      :short_name: smartCardLogon
    profiles:
      [profile]
certwriter:
  path: /home/user/r509-ca/certs
```

Listing E.1: r509-Konfigurationsdatei

Extension-Datei: Zertifikat mit mehreren Domainnamen im Subject Alternative Name

```
server_multidomain:
  basic_constraints:
    :critical: false
    :ca: false
  authority_info_access:
    :ocsp_location:
    -  :type: URI
       :value: http://ocsp.a-trust.at/ocsp
    :ca_issuers_location:
    -  :type: URI
       :value: http://www.a-trust.at/certs/a-sign-ssl-03.crt
  crl_distribution_points:
    :value:
    -  :type: URI
       :value: http://crl.domain.com/r509_howto_ca.crl
  certificate_policies:
    :value:
    - :policy_identifier: 1.2.40.0.17.1.20
      :cps_uris:
      - http://www.a-trust.at/docs/cp/a-sign-ss
      :user_notices:
  key_usage:
    :critical: true
    :value:
    - digitalSignature
```

```
    - keyEncipherment
  default_md: SHA1
  allowed_mds:
  - SHA1
  subject_item_policy:
    CN:
      :policy: required
    O:
      :policy: optional
    OU:
      :policy: optional
    ST:
      :policy: optional
    C:
      :policy: optional
    L:
      :policy: optional
```

Listing E.2: Zertifikatsformat Webserver Multidomain

Extension-Datei: Zertifikat mit UPN im Subject Alternative Name

```
client_upn:
  basic_constraints:
    :critical: false
    :ca: false
  authority_info_access:
    :ocsp_location:
    -  :type: URI
       :value: http://ocsp.a-trust.at/ocsp
    :ca_issuers_location:
    -  :type: URI
       :value: http://www.a-trust.at/certs/a-sign-Premium-Enc-02a.crt
  crl_distribution_points:
    :value:
    -  :type: URI
       :value: http://crl.domain.com/r509_howto_ca.crl
  certificate_policies:
    :value:
    - :policy_identifier: 1.2.40.0.17.1.12
      :cps_uris:
      - http://www.a-trust.at/docs/cp/a-sign-token
      :user_notices:
  key_usage:
    :critical: true
    :value:
    - digitalSignature
    - keyEncipherment
    - dataEncipherment
  extended_key_usage:
    :critical: false
    :value:
    - emailProtection
    - clientAuth
  default_md: SHA1
  allowed_mds:
  - SHA1
  subject_item_policy:
    CN:
      :policy: required
```

```
    O:
      :policy: optional
    OU:
      :policy: optional
    ST:
      :policy: optional
    C:
      :policy: optional
    L:
      :policy: optional
    SN:
      :policy: optional
    GN:
      :policy: optional
    T:
      :policy: optional
```

Listing E.3: Zertifikatsformat Client UPN

Extension-Datei: Zertifikate mit Qualified Certificate Statement

```
client_qc:
  basic_constraints:
    :critical: false
    :ca: false
  authority_info_access:
    :ocsp_location:
    -  :type: URI
       :value: http://ocsp.a-trust.at/ocsp
    :ca_issuers_location:
    -  :type: URI
       :value: http://www.a-trust.at/certs/a-sign-Premium-Enc-02a.crt
  crl_distribution_points:
    :value:
    -  :type: URI
       :value: http://crl.domain.com/r509_howto_ca.crl
  certificate_policies:
    :value:
    - :policy_identifier: 1.2.40.0.17.1.12
      :cps_uris:
      - http://www.a-trust.at/docs/cp/a-sign-Premium
      :user_notices:
    - :policy_identifier: 0.4.0.1456.1.1
  key_usage:
    :critical: true
    :value:
    - digitalSignature
    - nonRepudiation
  default_md: SHA1
  allowed_mds:
  - SHA1
  subject_item_policy:
    CN:
      :policy: required
    O:
      :policy: optional
    OU:
      :policy: optional
    ST:
      :policy: optional
```

```
    C:
      :policy: optional
    L:
      :policy: optional
    SN:
      :policy: optional
    GN:
      :policy: optional
    T:
      :policy: optional
  qc_statement:
    :value:
    - 30163008060604008e460101300a06082b06010505070b01
```

Listing E.4: Zertifikatsformat Qualified Certificate

Extension-Datei: Zertifikate mit Dienstleistereigenschaft

```
server_dienstleister:
  basic_constraints:
    :critical: false
    :ca: false
  authority_info_access:
    :ocsp_location:
    -  :type: URI
       :value: http://ocsp.a-trust.at/ocsp
    :ca_issuers_location:
    -  :type: URI
       :value: http://www.a-trust.at/certs/a-sign-ssl-03.crt
  crl_distribution_points:
    :value:
    -  :type: URI
       :value: http://crl.domain.com/r509_howto_ca.crl
  certificate_policies:
    :value:
    - :policy_identifier: 1.2.40.0.17.1.20
      :cps_uris:
      - http://www.a-trust.at/docs/cp/a-sign-ss
      :user_notices:
  key_usage:
    :critical: true
    :value:
    - digitalSignature
    - keyEncipherment
  default_md: SHA1
  allowed_mds:
  - SHA1
  subject_item_policy:
    CN:
      :policy: required
    O:
      :policy: optional
    OU:
      :policy: optional
    ST:
      :policy: optional
    C:
      :policy: optional
    L:
      :policy: optional
  dienstleister:
    :value:
```

```
45      - 0500
```

Listing E.5: Zertifikatsformat Webserver Dienstleistereigenschaft

F Befehle der Managementschnittstelle und Berechtigungen

Die nachfolgende Liste enthält die Befehle der Managementschnittstelle der CA-Software. Zusätzlich wird angegeben, wie die Autorisierung des jeweiligen Befehls durchzuführen ist. Abgesehen von dem Befehl „Anmeldung“ wird für jeden Befehl eine vorherige Anmeldung durch den Befehl „Anmeldung“ vorausgesetzt.

Befehl	**SO Unterschriften bestehendes System**	**SO Unterschriften neues System**
Anmeldung	1	1
AuditLog Suche	0	0
CA-Daten Laden	0	0
Zertifikatsformat erstellen	0[1]	2
Zertifikatsformat verändern	0[1]	2
Zertifikatsprozedur erstellen	2	2
Zertifikatsprozedur verändern	2	2
CA-Schlüssel erstellen	2	2
CA-Schlüssel verändern	2	2
CA anlegen	2	2
CA verändern	2	2
CA-Zertifikat erstellen	2	2
Officer suchen	0	0
Officer anlegen	2	2
Officer verändern	2	2
Verzeichnisdienst anlegen	0[1]	2
Verzeichnisdienst verändern	0[1]	2
Publication[2]anlegen	2	2
Publication[2]verändern	2	2
Zertifikat suchen	0	0

[1] Diese Einstellungen werden in Konfigurationsdateien vorgenommen, es werden keine SO-Unterschriften benötigt.

[2] Publications sind Regeln darüber, wo eine CRL bzw. ein Zertifikat veröffentlicht wird.

Die nachfolgende Tabelle zeigt die Befehle, die im neuen System zusätzlich implementiert werden.

Befehl	**SO-Unterschriften neues System**
CRL Erstellen	0
Product verändern	2
Officer anlegen Stapelverfahren	2
CA-Zertifikat verlängern	2
CA-Zertifikat widerrufen	2
CA-Gruppen anlegen	2
CA-Gruppen verändern	2
LDAP-Eintrag löschen	1
LDAP-Eintrag hinzufügen	1

G Größenberechnung der Sperrlisten

Als Basis für die Berechnung der Größe der Sperrlisten, dienen die in Tabelle G.1 angeführten Daten. Diese wurden aus dem aktuellen Produktivsystem der CA-Software entnommen.

CRL	Name	Anzahl Einträge	Größe [bytes]
CRL1	a-sign-premium-sig-05	0	779
CRL2	a-sign-corporate-light-03	117	4.749
CRL3	a-sign-SSL-03	411	16.389
CRL4	a-a-sign-premium-enc-03-SSL-03	87.989	3.168.136
CRL5	a-sign-premium-sig-03	107.684	3.877.156
CRL6	a-sign-token 03	21.580	777.400
CRL7	a-sign-premium-mobile-03	42.312	1.523.770

Tabelle G.1: Sperrlisten Anzahl Einträge und Größe

$$\text{Größe 1 Eintrag} = \frac{(\text{Größe gesamt} - \text{Größe Header})}{\text{Anzahl Einträge}}$$

CRL1: $\text{Größe Header} = 779\text{bytes}$

$$\text{CRL2: Größe 1 Eintrag} = \frac{(4.749\text{bytes} - 779\text{bytes})}{117} = 33,93\text{bytes}$$

$$\text{CRL3: Größe 1 Eintrag} = \frac{(16.389\text{bytes} - 779\text{bytes})}{411} = 37,98\text{bytes}$$

$$\text{CRL4: Größe 1 Eintrag} = \frac{(3.168.136\text{bytes} - 779\text{bytes})}{87.989} = 36,0\text{bytes}$$

$$\text{CRL5: Größe 1 Eintrag} = \frac{(3.877.156\text{bytes} - 779\text{bytes})}{107.684} = 36,0\text{bytes}$$

$$\text{CRL6: Größe 1 Eintrag} = \frac{(777.400\text{bytes} - 779\text{bytes})}{21.580} = 35,99\text{bytes}$$

$$\text{CRL7: Größe 1 Eintrag} = \frac{(1.523.770\text{bytes} - 779\text{bytes})}{42.312} = 35,99\text{bytes}$$

$$\text{Mittelwert 1 Eintrag} = \frac{33,93 + 37,98 + 36,0 + 36,0 + 35,99 + 35,99\ [\text{bytes}]}{6} = 35,98\text{bytes}$$

$$\text{Mittelwert 1 Eintrag} \approx 36\text{bytes}$$

In weiterer Folge ergibt sich daraus die folgende Formel zur Berechnung der Größe einer Sperrliste:

$$\text{CRL Größe} \approx (\text{Anzahl Einträge} * 36\text{bytes}) + 779\text{bytes}$$

H Inhalt der beigelegten CD-ROM

```
cd/
├── Analyse/
│   ├── ejbca/
│   ├── openca/
│   ├── OpenSSL_CA/
│   ├── r509/
│   ├── simpleAuthority/
│   ├── Symantec_Managed_PKI_Serivces/
│   ├── XCA_Zertifikate/
│   └── Zertifikate_Vorlagen/
├── CRL_Größenprobleme/
│   └── CrlToCsv/
├── Softwareentwicklung/
│   ├── BouncycastleCSharp/
│   ├── BouncycastleJava/
│   ├── IAIK/
│   ├── JavaSunSecurityCertAndKeyGen/
│   ├── MsCryptoApi/
│   ├── OpenSSL/
│   └── pyOpenSSL/
├── Softwaretests/
│   └── Fuzzing/
└── Tools/
    ├── dumpasn1/
    └── OpenSSL/
```

I Zeitaufzeichnung

Stunden		Kapitel
1		Einleitung
85		Stand der Technik
	14	Allgemein
	2	Smarttrust CA
	13	EJBCA PKI CA
	12	XCA - X Certificate and key management
	7	SimpleAuthority
	6	OpenSSL-CA
	10	Microsoft-CA
	8	OpenCA PKI/OpenXPKI
	4	Symantec Managed PKI Certificate Service
	2	TinyCa
	1	gnoMint
	2	pyCa
	4	PHP-CA
9		Präzisierung der Aufgabenstellung
57		Systemkonzept
	16	Komponentenaufteilung
	10	Ausfallszenarien/Lastverteilung
	10	Zertifikatsformat und -regeln
	14	Sperrlisten (CRL)
	7	OCSP-Server
49		Softwareentwicklung
	7	Allgemein
	9	OpenSSL-CA
	8	Bouncycastle C#
	8	Bouncycastle Java
	9	Microsoft CryptoAPI
	5	IAIK
	3	Java Sun Security CertAndKeyGen
8		Testszenarien
5		Zusammenfassung der Ergebnisse und Ausblick

Literaturverzeichnis

[AD07] A. Deacon, R. Hurst: *The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments*. Technischer Bericht 5019, September 2007. https://www.ietf.org/rfc/rfc5019.txt.

[Bac09] Bachfeld, Daniel: *Attacken auf SHA-1 weiter vereinfacht*, 2009. http://www.heise.de/security/meldung/Attacken-auf-SHA-1-weiter-vereinfacht-180587.html, Accessed: 2014-06-27.

[Bra07] Bradner, S.: *Key words for use in RFCs to Indicate Requirement Levels*. Technischer Bericht 2119, März 2007. https://www.ietf.org/rfc/rfc2119.txt.

[CA10] Carlisle Adams, Steve Lloyd: *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison Wesley Pub Co Inc, 2010.

[Cal11] Calderone, Jean Paul: *Welcome to pyOpenSSL's documentation! - pyOpenSSL 0.14 documentation*, 2011. http://pythonhosted.org//pyOpenSSL/, Accessed: 2014-06-21.

[CBK10] Carsten B. Kinder, Mark B. Cooper: *Failover Clustering and Active Directory Certificate Services in Windows Server 2008 and Windows Server 2008 R2*, Januar 2010. http://gallery.technet.microsoft.com/Failover-Clustering-and-b3ea8858.

[CGC11] Coronado-García, L.C und Pérez Leguízamo C.: *A misson-critical certificate authority architecture for high reliability and response time*. International Journal of Critical Computer-Based Systems, Vol. 2:Seite 6–24, 2011.

[DC08] D. Cooper, S. Santesson, S. Farrell S. Boeyen R. Housley W. Polk: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Technischer Bericht 5280, Mai 2008. https://www.ietf.org/rfc/rfc5280.txt.

[EJB14a] EJBCA team: *EJBCA - Open Source PKI Certificate Authority - Admin Guide*, Juli 2014. http://www.ejbca.org/docs/adminguide.html#Custom%20Certificate%20Extensions, Accessed: 2014-07-15.

[EJB14b] EJBCA team: *EJBCA - Open Source PKI Certificate Authority - Home*, Juni 2014. http://www.ejbca.org/, Accessed: 2014-07-15.

[EJB14c] EJBCA team: *EJCBA - Sample setup architecture*, Juni 2014. http://www.ejbca.org/docs/architecture.html, Accessed: 2014-07-15.

[Fou06] Foundation, Free Software: *GNU Crypto - GNU Project - Free Software Foundation (FSF)*, November 2006. http://www.gnu.org/software/gnu-crypto/, Accessed: 2014-06-20.

[Gro05] Group, OpenCA: *OpenCA Guide for Versions 0.9.2+*. Website, August 2005. http://www2.openxpki.org/docs/guide/openca-guide.html.

[Hag13] Hagelkruys, Patrick: *Design und Konzeption einer Softwarelösung zur Erstellung und Verwaltung von digitalen Zertifikaten*, August 2013. Projektarbeit, Hochschule Mittweida, Fakultät EIT.

[Hoh13] Hohnstaedt, Christian: *xca | Free Security & Utilities software downloads at SourceForge.net*, März 2013. http://sourceforge.net/projects/xca/, Accessed: 2014-05-12.

[Lit13] Litzenberger, Dwayne C.: *Python Cryptography Toolkit (pycrypto)*, Oktober 2013.

[LLC14] LLC, Jax Systems: *sun.security.x509: public final class: CertAndKeyGen*, 2014. http://www.docjar.com/docs/api/sun/security/x509/CertAndKeyGen.html, Accessed: 2014-06-20.

[Mac13] MacWilliams, Justin: *Chandler Wiki : Me Too Crypto*, August 2013. https://web.archive.org/web/20130825211205/http://chandlerproject.org/Projects/MeTooCrypto, Accessed: 2014-06-21.

[Mic09] Microsoft: *Failover Clustering in Windows Server 2008 R2 White Paper*, April 2009. http://download.microsoft.com/download/F/2/1/F2146213-4AC0-4C50-B69A-12428FF0B077/WS08%20R2%20Failover%20Clustering%20White%20Paper.doc.

[Mic13] Microsoft: *Defining CA Types and Roles: Public Key*, Mai 2013. http://technet.microsoft.com/en-us/library/cc756989(v=ws.10).aspx, Accessed: 2014-05-18.

[Mic14a] Microsoft: *Cryptography Reference*, 2014. http://msdn.microsoft.com/en-us/library/windows/desktop/aa380256(v=vs.85).aspx, Accessed: 2014-06-19.

[Mic14b] Microsoft: *Performance and Reliability Patterns*, 2014. http://msdn.microsoft.com/en-us/library/ff648802.aspx, Accessed: 2014-05-30.

[MM99] M. Myers, R. Ankney, A. Malpani S. Galperin C. Adams: *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. Technischer Bericht 2560, Juni 1999. https://www.ietf.org/rfc/rfc2560.txt.

[MM13] M. Myers, R. Ankney, A. Malpani S. Galperin C. Adams: *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. Technischer Bericht 6960, Juni 2013. https://www.ietf.org/rfc/rfc6960.txt.

[Ofe06] Ofenschüßel, Heinz: *A-Trust Certification Authority Functional Specification*. Hewlett Packard Ges.m.b.H, Mai 2006.

[Ope14a] OpenSSL: *OpenSSL: Documents, x509v3_ config(5)*, Juli 2014. https://www.openssl.org/docs/apps/x509v3_config.html, Accessed: 2014-07-15.

[Ope14b] OpenSSL: *OpenSSL: The Open Source toolkit for SSL/TLS*, Juni 2014. https://www.openssl.org/, Accessed: 2014-07-15.

[Pyl09] Pyle, Ned: *Designing and Implementing a PKI*. Microsoft's official enterprise support blog for AD DS and more, September 2009. http://blogs.technet.com/b/askds/archive/2009/09/01/designing-and-implementing-a-pki-part-i-design-and-planning.aspx.

[RSA09] RSA Security Inc.: *PKCS #11 Base Functionality v2.30: Cryptoki – Draft 4*, April 2009. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30b-d6.pdf.

[RWS02] RWS: *SmartTrust Certificate Manager - Technical Description*. April 2002.

[Sab14] Sabet, Ramin: *CA Abnahme*. A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH, April 2014.

[Sch96] Schneier, Bruce: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Band Second Edition. John Wiley & Sons, 1996.

[Sch13] Schwemmer: *Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen)*, Februar 2013. http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/

QES/Veroeffentlichungen/Algorithmen/2013Algorithmenkatalog.pdf?__blob=publicationFile&v=1.

[S.d07] S.dergren, Rolf: *CA Administrator's Guide*. nexus, Dezember 2007.

[Sei09] Seitz, Justin: *Hacking mit Python*, Band First Edition. dpunt.verlag GmbH, 2009.

[Ser06] Sermersheim, Dd. J.: *Lightweight Directory Access Protocol (LDAP): The Protocol*. Technischer Bericht 4511, Juni 2006. https://www.ietf.org/rfc/rfc4511.txt.

[SF10] S. Farrell, R. Housley, S. Turner: *An Internet Attribute Certificate Profile for Authorization*. Technischer Bericht 5755, Januar 2010. https://www.ietf.org/rfc/rfc5755.txt.

[SIC14] SIC, Stiftung: *Secure Information and Communication Technologies / Home - Stiftung SIC*, 2014. https://jce.iaik.tugraz.at/, Accessed: 2014-06-20.

[Sig08] *Bundesrecht konsolidiert: Gesamte Rechtsvorschrift für Signaturverordnung 2008*, 2008. http://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20005618, Accessed: 2014-07-24.

[Sim14] SimpleAuthority: *SimpleAuthority - The Simple Certification Authority*, 2014. http://simpleauthority.com/, Accessed: 2014-05-14.

[SS01] S. Santesson, W. Polk, P. Barzin M. Nystrom: *Internet X.509 Public Key Infrastructure Qualified Certificates Profile*. Technischer Bericht 3039, Januar 2001. https://www.ietf.org/rfc/rfc3039.txt.

[Sta01] Stark, Shubhangi: *Sichere IT-Kommunikation über unsichere Netze*. Diplomarbeit, Ruprecht-Karls-Universität Heidelberg, April 2001.

[Str99] Ströder, Michael: *Einführung kryptographischer Techniken zur gesicherten Nutzung des Internet bei der Propack Data GmbH*. Diplomarbeit, Universität Karlsruhe (TH), Januar 1999. http://web.archive.org/web/20031211040920/http://www.stroeder.com/DA/.

[Str03a] Ströder, Michael: *pyCA - X.509 CA*, April 2003. http://www.pyca.de/, Accessed: 2014-05-15.

[Str03b] Ströder, Michael: *pyCA - X.509 CA*, April 2003. http://www.pyca.de/config.html#openssl.cnf, Accessed: 2014-05-15.

[Toi13] Toivonen, Heikki: *M2Crypto*, April 2013.

[Wik14] Wikipedia: *YAML — Wikipedia, The Free Encyclopedia*, Juli 2014. http://en.wikipedia.org/wiki/YAML, Accessed: 2014-07-24.

Glossar

API Application Programming Interface, Programmierschnittstelle.

authorityInfoAccess Zertifikatserweiterung: Authority Information Access.

CryptoAPI Cryptographic Application Programming Interface, eine Programmierschnittstelle von Microsoft für Kryptographiefunktionen.

DoS-Angriff Denial-of-Service-Angriff.

extKeyUsage Zertifikatserweiterung: Extended Key Usage.

Fuzzer Programm, das ein Fuzzing durchführt, siehe Fuzzing.

Fuzzing Testmethode, bei der zufällige fehlerhafte Eingangsdaten generiert werden.

givenName Zertifikatserweiterung: given name, Vorname.

HSM Hardware Security Module, Kryptographie-Hardware für die Speicherung von nicht exportierbaren privaten Schlüssel.

IAIK Institute for Applied Information Processing and Communications, Institut der TU Graz.

LDAP Lightweight Directory Access Protocol, Protokoll des Zugriffs auf Verzeichnisse über TCP/IP-Kommunikation.

RA Registration Authority, Registrierungsstelle.

SO Security Officer, Sicherheitspersonal bei A-Trust, das Änderungen an der CA im Vieraugenprinzip durchführen darf.

Sourceforge Hosting Provider für Open-Source-Projekte, `http://sourceforge.net/`.

subjectAltName Zertifikatserweiterung: Subject Alternative Name.

YAML YAML Ain' t Markup Language (früher: Yet Another Markup Language), ist ein Daten-Serialisierungs-Format basierend auf den Konzept von div. Programmiersprachen, XML und MIME (s. [Wik14]).

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 9. Juli 2014